Net-Centric Implementation

Part 5: Developer Guidance

v3.3.0

11 October 2011

Net-Centric Enterprise Solutions for Interoperability (NESI) is a collaborative activity of the USN PEO for C4I and Space, the USAF Electronic Systems Center, and the Defense Information Systems Agency.

Approved for public release; distribution is unlimited.

SSIC: 3093.4

NESI-X Subdocument generated using: **View, Part 5: Developer Guidance**
Generated: **Wed Oct 26 13:37:08 PDT 2011**
NESI-X Version: **v2.9.6 build 7130 - 2011/09/22 09:41**

Table of Contents

P1117: NESI Executive Summary

Net-Centric Enterprise Solutions for Interoperability (***NESI***) provides actionable guidance for acquiring net-centric solutions that meet DoD **Network Centric Warfare** goals. The concepts in various directives, policies and mandates, such as those included in the References section of this perspective, are the basis of NESI guidance. The NESI *Net-Centric Implementation* documentation does the following: addresses architecture, design and implementation; provides compliance checklists; and includes a collaboration environment with a repository.

NESI is a body of architectural and engineering knowledge that helps guide the design, implementation, maintenance, evolution, and use of **Information Technology** (**IT**) in net-centric solutions for military application. NESI provides specific technical recommendations that a DoD organization can use as references. NESI serves in many areas as a reference set of compliant instantiations of DoD directives, policies and mandates.

NESI is derived from a studied examination of enterprise-level needs and from the collective practical experience of recent and on-going program-level implementations. NESI is based on current and emergent technologies and describes the practical experience of system developers within the context of a minimal top-down technical framework. NESI guidance strives to be consistent with commercial best practices in the area of enterprise computing and IT.

NESI applies to all phases of the acquisition process as defined in DoD Directive 5000.1 [R1164] and DoD Instruction 5000.2; [R1165] NESI provides explicit guidance for implementing net-centricity in new acquisitions and for migrating legacy systems to greater degrees of net-centricity.

NESI subsumes a number of references and directives; in particular, the Air Force C2 Enterprise Technical Reference Architecture (C2ERA) and the Navy Reusable Applications Integration and Development Standards (RAPIDS). Initial authority for NESI is per the Memorandum of Agreement between Commander, Space and Naval Warfare Systems Command (SPAWAR); Navy Program Executive Officer, C4I & Space (now PEO C4I); and the United States Air Force Electronic Systems Center (ESC), dated 22 December 2003, Subject: Cooperation Agreement for Net-Centric Solutions for Interoperability (NESI). The Defense Information Systems Agency (DISA) formally joined the NESI effort in 2006.

Perspectives	NESI ***Perspectives*** describe a topic and encompass related, more specific Perspectives or encapsulate a set of Guidance and Best Practice details, Examples, References, and Glossary entries that pertain to the topic.
Guidance	NESI ***Guidance*** is in the form of atomic, succinct, absolute and definitive Statements related to one or more Perspectives. Each Guidance Statement is linked to Guidance Details which provide Rationale, relationships with other Guidance or Best Practices, and Evaluation Criteria with one or more Tests, Procedures and Examples which facilitate validation of using the Guidance through observation, measurement or other means. Guidance Statements are intended to be binding in nature, especially if used as part of a Statement of Work (SOW) or performance specification.
Best Practices	NESI ***Best Practices*** are advisory in nature to assist program or project managers and personnel. Best Practice Details can have all the same parts as NESI Guidance. The use of NESI Best Practices are at the discretion of the program or project manager.
Examples	NESI ***Examples*** illustrate key aspects of Perspectives, Guidance, or Best Practices.
Glossary	NESI ***Glossary*** entries provide terms, acronyms, and definitions used in the context of NESI Perspectives, Guidance and Best Practices.
References	NESI ***References*** identify directives, instructions, books, Web sites, and other sources of information useful for planning or execution.

Releasability Statement

NESI *Net-Centric Implementation* v3.3 is cleared for public release by competent authority in accordance with DoD Directive 5230.9; [R1232] *Distribution Statement A: Approved for public release; distribution is unlimited* applies to the documentation set. Obtain electronic copies of this document at http://nesipublic.spawar.navy.mil.

Vendor Neutrality

NESI documentation sometimes refers to specific vendors and their products in the context of examples and lists. However, NESI is vendor-neutral. Mentioning a vendor or product is not intended as an endorsement, nor is a lack of mention intended as a lack of endorsement. Code examples typically use open-source products since NESI is built on the open-source philosophy. NESI accepts inputs from multiple sources so the examples tend to reflect contributor preferences. Any products described in examples are not necessarily the best choice for every circumstance. Users are encouraged to analyze specific project requirements and choose tools accordingly. There is no need to obtain, or ask contractors to obtain, the tools that appear as examples in this guide. Any lists of products or vendors are intended only as examples, not as a list of recommended or mandated options.

Disclaimer

Every effort has been made to make NESI documentation as complete and accurate as possible. Even with frequent updates, this documentation may not always immediately reflect the latest technology or guidance. Also, references and links to external material are as accurate as possible; however, they are subject to change or may have additional access requirements such as Public Key Infrastructure (PKI) certificates, Common Access Card (CAC) for user identification, and user account registration.

Contributions and Comments

NESI is an open project that involves the entire development community. Anyone is welcome to contribute comments, corrections, or relevant knowledge to the guides via the Change Request tab on the NESI Public site, http://nesipublic.spawar.navy.mil, or via the following email address: nesi@spawar.navy.mil.

P1118: Part 5: Developer Guidance

Part 5: Developer Guidance provides program managers, engineers and software developers detailed implementation guidance for applications, services, and data. This effort leverages current best practices from the software development community to enable the **Department of Defense** (**DoD**) to create net-centric, extensible and scalable enterprise solutions. The goal is to modernize and improve the development of net-centric **applications** and **services** as critical warfighter capabilities. The standards, policies, and processes within Part 5 are useful for building and maintaining applications and services that must interoperate in the DoD Net-Centric Enterprise.

Part 5 provides software development and architecture guidance, best coding practices, lessons learned, and guidance. It serves as a reference resource in support of specific topics, not a document to read in a sequential, linear fashion. The guidance in Part 5 is useful in a variety of ways including the following:

- Supporting modular software development to minimize risk and impacts of changes to application developers
- Migrating legacy systems to the net-centric environment (in conjunction with Part 3: Migration Guidance [P1198])
- Implementing connection strategies that extend the life and reach of legacy applications while legacy application developers restructure their systems
- Evaluating software deliverables for net-centricity and interoperability

Detailed Perspectives

- Implement a Component-Based Architecture [P1034]
- Public Interface Design [P1060]
- Standard Interface Documentation [P1069]
- Automate the Software Build Process [P1007]
- Programming Languages [P1113]
- Software Security [P1065]
- Data [P1012]
- User Interfaces [P1058]
- Middleware [P1052]
- Source Code Migration to Support IPv4 and IPv6 [P1396]
- Logging [P1448]

P1034: Implement a Component-Based Architecture

The Federation of Government Information Processing Councils/Industry Advisory Council (FGIPC/IAC) defined **Component-Based Architecture** (**CBA**) as follows in a March 2003 paper titled Succeeding with "Component-Based Architecture in e-Government":

"An architecture process that enables the design of enterprise solutions using pre-manufactured components. The focus of the architecture may be a specific project or the entire enterprise. This architecture provides a plan of what needs to be built and an overview of what has been built already." [Succeeding with Component-Based Architecture]

CBA represents a shift from the traditional, custom-development-oriented, "design, code, and test" approach that has been used throughout the DoD in the past to a more business-oriented "architect, acquire, and assemble" approach.

The custom-development approach has been successful in building many systems. However, the integration, evolution, reuse and cost of these systems have presented a problem. Consequently, these custom-developed systems have been labeled as archaic **stovepipes** that can not plug-and-play with other systems.

CBA promises benefits such as shorter time to market, lower risk, and modular and adaptive systems.

The core of CBA is components. The NESI definition of the term **component** is that it is one of the parts that make up a system; a component may be hardware or software and may be subdivided into other components. The following guidance statements capture the essence of components.

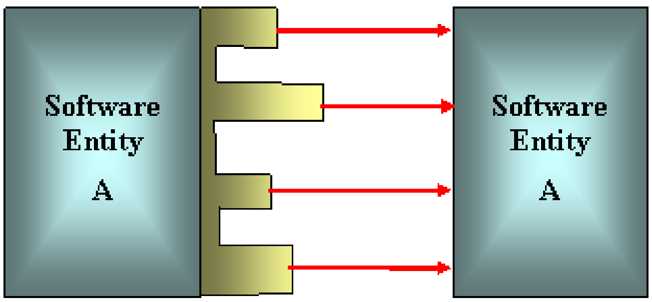
Guidance

- G1011: Make **components** independently deployable.
- G1012: Use a set of services to expose **component** functionality.
- G1217: Develop and use externally configurable components.

Part 5: Developer Guidance > Public Interface Design

P1060: Public Interface Design

A public interface is the logical point at which independent software entities interact. The entities may interact with each other within a single computer, across a network, or across a variety of other topologies. It is important that public **interfaces** be stable and designed to support future changes, enhancements, and **deprecation** in order for the interaction to continue.



I1007

Guidance

- G1001: Use formal standards to define public **interfaces**.
- G1002: Separate public **interfaces** from implementation.
- G1003: Separate shared **Application Programming Interfaces** (**APIs**) from internal APIs.
- G1004: Make public **interfaces** backward-compatible within the constraints of a published **deprecation** policy.
- G1008: Isolate the Web service portlet from web hosting infrastructure dependencies by using the **Web Services for Remote Portlets** (**WSRP**) Specification protocol.
- G1010: Use a logging facade that allows for specifying the underlying logging framework during software deployment.
- G1018: Assign version identifiers to all public interfaces.
- G1019: Deprecate public interfaces in accordance with a published deprecation policy.
- G1022: Insulate public **interfaces** from compile-time dependencies.
- G1073: Isolate vendor extensions to **enterprise service** interfaces.
- G1208: Add new functionality rather than redefining existing interfaces in a manner that brings incompatibility.
- G1213: Provide an architecture design document.
- G1214: Provide a document with a plan for **deprecating** obsolete **interfaces**.
- G1215: Provide a coding standards document.
- G1216: Provide a software release plan document.

Best Practices

- BP1007: Develop software using **open standard Application Programming Interfaces** (**APIs**).
- BP1021: Create fully encapsulated classes.
- BP1240: Present complete and coherent sets of concepts to the user.
- BP1241: Design statically typed **interfaces**.
- BP1242: Minimize an **interface's** dependencies on other interfaces.
- BP1243: Express **interfaces** in terms of application-level types.

P1069: Standard Interface Documentation

This section provides guidance for documenting source code. The references provide links on documenting code for the Java and the Microsoft .NET environments. For all other languages, configuration files, and XML files, please follow the associated language-specified format for documentation.

Javadoc Commands

The **Javadoc** tool parses special tags when they are embedded within a Javadoc comment. These doc tags enable a programmer to autogenerate a complete, well-formatted API from the source code. The tags start with an ampersand (@) and are case-sensitive; an "a" is different from an "A."

A tag must start at the beginning of a line, after any leading spaces and an optional asterisk, or it will be treated as normal text. By convention, group tags with the same name together. For example, put all **`@see`** tags together.

Guidance

- G1027: Internally document all source code developed with Department of Defense (DoD) funding.

P1007: Automate the Software Build Process

A software build process interfaces with source control, compiles code, creates executables, runs unit tests, packages and deploys, and generates documentation. Automating a software build process provided for the following advantages:

- provides for improved quality, consistency, and repeatability as the software is built in the same manner each time
- reduces the time required to compile, link, and package software components allowing for more often build cycles
- helps to reduce dependencies on key personnel for building software
- supports faster integration cycles as the time required to build, link, and package software is reduced

In addition, using an automated build process that is executed outside of the **Integrated Development Environments** (**IDEs**) reduces problems when sharing code between groups using different IDEs.

Common Build Tools

There are many products available that support automating the software build process. The guidance provided below aids in selecting tools which provide the most interoperability of build processes and prevents dependencies on any given IDE. The following are examples of software build tools in use with the software development industry:

Make is a commonly used name (for which there are various implementations such as GNU Make) for a build tool that compiles and links source code. Make conducts build operations according to a build instruction contained in a file called the makefile. Make is commonly used for building software based on C and C++, although it is applicable to most programming languages and development environments. Tools such as Automake (for which there are various implementations such as GNU Automake) provide support for building makefiles.

Like Automake, the Makefile, Project, and Workspace Creator (MPC) supports the creation of makefiles for the Make tool as well as supports creating build files for several other commercial IDEs as well.

Apache Ant and Apache Maven are common multi-platform build tools used to automate the build process for Java software development. Both are based on the use of XML for describing the build process. Build processes are generally described procedurally in Apache Ant and are described more declaratively in Apache Maven.

Guidance

- G1190: Use a build tool.
- G1218: Use a build tool that supports operation in an automated mode.
- G1219: Use a build tool that checks out files from configuration control.
- G1220: Use a build tool that **compiles** source code and dependencies that have been modified.
- G1221: Use a build tool that creates libraries or archives after all required compilations are complete.
- G1222: Use a build tool that creates executables.
- G1223: Use a build tool that is capable of running unit tests.
- G1224: Use a build tool that cleans out intermediate files that can be regenerated.
- G1225: Use a build tool that is independent of the **Integrated Development Environment**.

P1113: Programming Languages

This Complex Perspective contains a collection of Detailed Perspectives which provide programming language guidance. The purpose of the following Perspectives is to provide language-specific guidance with the purpose of improving interoperability and net-centricity.

Detailed Perspectives

- C++ [P1090]
- VHDL [P1088]

Part 5: Developer Guidance > Programming Languages > C++

P1090: C++

The development of software is a complex and difficult process that covers a wide range of activities starting at the earliest phases of requirements analysis all the way through the release of the software. In the **DoD**, many formal processes, documents and reviews need to occur before software is ready for release as a product. This complexity has increased as the accepted software development processes has evolved to embrace Object-Oriented techniques and incremental development.

A number of individuals, institutions, companies and products have attempted to solve software development issues and have produced a number of very useful papers, dissertations and books. It is not the intent of this NESI perspective to re-state written material or to endorse any particular institution, corporation or product. This perspective highlights those practices relating to the use of the C++ language which have demonstrated an ability to increase interoperability and enable net-centricity. In particular, one goal of this perspective is to identify guidance and best practices which facilitate interoperability of C++ code in order to promote reuse. Interoperability and code reuse depends on security and trustability; the Carnegie Mellon University Computer Emergency Response Team (CERT) [R1301] provides additional detail regarding secure programming practices for C++.

This perspective includes three sub-perspectives; much of the content is modeled after coding standards Herb Sutter and Andrei Alexandrescu put forth in the referenced text.[R1150]

Detailed Perspectives

- C++ Namespaces and Modules [P1115]
- C++ Operator Overloading [P1114]
- C++ Header Files [P1089]

Part 5: Developer Guidance > Programming Languages > C++ > C++ Namespaces and Modules

P1115: C++ Namespaces and Modules

Namespaces and **modules** are abstract containers for related items. Often, software developers use both to isolate related items in order to promote reuse. Namespaces provide a context within which to define identifiers (i.e., classes, constants, variables, and functions). One advantage of namespaces is that they allow multiple identifiers with the same name to be used in the same code without name collisions.

Guidance

- G1778: Place all **`#include`** statements before all namespace **`using`** statements.
- G1779: Explicitly namespace-qualify all names in header files.

Best Practices

- BP1781: Allocate and de-allocate all **module** objects within the module that contains the objects.
- BP1782: Do not propagate exceptions across **module** boundaries.
- BP1783: Use portable types in a **module's** interface.

P1114: C++ Operator Overloading

C++ allows for overloading of operators in order to change their implementation depending on the type of arguments provided. This can improve code clarity and serve as a short hand for developers. However, developers must be careful to not change the expected behavior or semantics of an operator in a way that provides unexpected behavior to developers using the code. Code which has clearly understood behavior has a better chance of being reusable.

Guidance

- G1775: Do not overload the logical **`AND`** operator.
- G1776: Do not overload the logical **`OR`** operator.
- G1777: Do not overload the **`comma`** operator.

Best Practices

- BP1780: Only overload arithmetic operators for objects that are arithmetic in nature.

P1089: C++ Header Files

A header file in C++ describes the interface of the related implementation file. Header files serve as a communication mechanism to describe interfaces including data-types, **namespaces**, required resources, as well as serving as a source of reference documentation. The compiler uses header files during compilation, and humans use header files during software development. To promote reuse, header files need to be self-describing and developed such that compilation is straight forward and consistent from one compile to another.

Guidance

- G1773: Use **`#include`** guards for all headers.
- G1774: Make header files self-sufficient.
- G1779: Explicitly namespace-qualify all names in header files.

Part 5: Developer Guidance > Programming Languages > VHDL

P1088: VHDL

The development of hardware described by software is a complex and difficult process that covers a wide range of activities: starting at the earliest phases of requirements analysis all the way through the fabrication of a functioning digital circuit. One language developed for describing digital circuits is **Very High Speed Integrated Circuit** (**VHSIC**) Hardware Description Language (**VHDL**).

In the **DoD**, there are many formal processes, documents and reviews which need to be done in order for the software code to be approved to be developed into a physical circuit. This complexity has been made more complicated in nature as modern chip designs have become increasingly large and intricate. There have been many articles and books written on these issues. It is not the intent of this perspective to re-state written material. It is the intent of this perspective to highlight those practices which have been demonstrated to increase interoperability and reuse of VHDL code.

Detailed Perspectives

- VHDL Coding and Design [P1091]
- VHDL Testbench [P1094]
- VHDL Synthesizable Design [P1093]
- VHDL Synchronous Design [P1092]

P1091: VHDL Coding and Design

There are coding and design decisions that are made during the lifecycle of a program or project which can have significant impact on interoperability and net-centricity. Many of these decisions directly relate to **cohesion** and **coupling**. Modifications to a project's code often create additional obstacles and decreases efficiency. The purpose of this perspective is to provide guidance and best practices to minimize these problems.

Guidance

- G1717: Use constants instead of hard-coded numbers for characteristics that may change throughout the lifetime of the model.

Best Practices

- BP1720: Do not use commonly predefined VHDL identifier names for other identifiers.
- BP1721: Define a VHDL package for closely related VHDL items that support an application function.
- BP1722: Employ VHDL components for commonly used VHDL described circuits.

P1094: VHDL Testbench

A **VHDL** testbench is a **VHDL component** used to verify that a developing circuit design is functioning as planned. The testbench generates the stimulus to drive the unit under test under a variety of test conditions, verifies that it meets specifications, and reports all errors and warnings in a concise human readable format. The testbench is used during the simulation phase of digital electronic design automation.

Guidance

- G1719: Automate testbench error checking in VHDL development.

P1093: VHDL Synthesizable Design

To be able physically to implement hardware described by software, the design must be synthesizable. Synthesis is a process where an abstract form of described circuit behavior (e.g., **VHDL** code) is mapped to an implementation in terms of logic gates (**`AND`**, **`OR`**, **`NOT`**, etc.). Logic synthesis is an essential part of digital electronic design automation and is often the step following code compilation and simulation.

Best Practices

- BP1723: Do not use guarded signals.

P1092: VHDL Synchronous Design

The engineers of digital integrated circuits (ICs) are very careful to make sure their designs are correct, for it is imperative that hardware designs are correct before being fabricated into physical circuits. However, digital circuits are not easily testable and real tests cannot be done on them until the circuit design has been finalized and physically produced. This is one of the reasons why the majority of today's digital designs are based on a synchronous design to improve the probability that the final produced chip will work by simplifying the process and using reliable techniques.

Guidance

- G1718: Design circuits to be synchronous.

P1065: Software Security

Security is a top priority in the nation's agenda. It is more critical than ever to establish security guidelines for new and evolving military systems, especially for information technology based systems. Software vulnerabilities, malicious code, and software that does not perform as intended pose an increased risk to the loss of operational capability and information superiority.

Software, in order to be useful, must be dependable (executes predictably and correctly under all conditions, including hostile conditions), trustworthy (contains few vulnerabilites or weaknesses that allow intentional loss of dependability or malicious behaviour of the software), and survivable (resilient to attack and able to recover quickly with minimal damages or loss of data from attacks it cannot resist). At a minimum, good secure software provides the following:

- **Identification**, **Authentication**, and **Authorization** to ensure proper control of access to the software and the data it handles
- **Confidentiality** to prevent unintended disclosure of information
- **Integrity** to ensure correctness and reliability of the software along with **information assurance** to provide assertions that the software, and the data handled by it, are used correctly
- Availabiliy to ensure the software is able to be used when required
- Management capabilities to manage and audit the use of the software

Software security requires active consideration throught the lifecycle to include the requirements, development, deployment, operation, and substainment phases.

The detailed perspectives listed below provide guidance for the development of secure software organized around two security aspects that apply to the development of any software system. The first aspect is the technologies and standards used to enable security, and the second is the policies and processes which promote security.

The following resources provide additional information to supplement the more specific content of the items linked in the Detailed Perspectives subsection.

- The Information Assurance Technology Analysis Center (IATIC) State-of-the-Art Report *Software Security Assurance* [R1338] provides techniques (to include process models, life cycle models, and best practices) useful for producing secure software.
- The Software Assurance Acquisition Working Group report *Software Assurance in Acquisition: Mitigating Risks to the Enterprise* [R1340] provides processes and guidance useful for both software practitioners and acquisition personel to ensure the development of software that is secure.
- The National Institute of Standards and Technology (NIST) Special Publication (SP) 800-117, *Guide to Adopting and Using the Security Content Automation Protocol (SCAP)*, and SP 800-126, *The Technical Specification for the Security Content Automation Protocol (SCAP)*, provide information on a suite of specifications that standardize the format and nomenclature by which security software products communicate software flaw and security configuration information. Both of these Special Publications are available via the NIST *Special Publications (800 Series)* index.[R1355] Software developers can use SCAP to make security settings available through automation.

Detailed Perspectives

- Technologies and Standards for Implementing Software Security [P1391]
- Policies and Processes for Implementing Software Security [P1392]

P1391: Technologies and Standards for Implementing Software Security

The following perspectives provide guidance and best practices regarding the role of technologies and standards for implementing software security in the following areas:

- Using Public Key Infrastructure (PKI) related technologies to enable **identification**, **authentication**, and **authorization**
- Using XML Digital Signatures to provide non-repudiation
- Using encryption technologies and guidance to provide confidentiality
- Providing secure services
- Protecting data storage
- Using programming languages securely

Detailed Perspectives

- Public Key Infrastructure (PKI) and PK Enable Applications [P1061]
- Key Management [P1041]
- Certificate Processing [P1009]
- Smart Card Logon [P1315]
- XML Digital Signatures [P1387]
- Encryption Services [P1020]
- SOAP Security [P1085]
- Security Assertion Markup Language (SAML) [P1189]
- RDBMS Security [P1064]
- LDAP Security [P1042]
- JNDI Security [P1039]
- Application Resource Security [P1005]
- Java Security [P1038]

P1061: Public Key Infrastructure (PKI) and PK Enable Applications

More and more secure **client**/**server** applications are appearing on the market. Applications today are relying heavily on **Digital Signature** technology to certify messages received were indeed sent by the sender. Both of these technologies use **Public Key encryption**, which is currently the only feasible way of implementing security over an insecure network such as the **NIPRNet**. Public Key encryption ensures that any form of communication that many contain sensitive information (i.e., passwords, credit card numbers) is protected while in transit and provides assurance to the receiver that the message was really sent by the sender. In the case of Web-based technologies, this is accomplished with a server that implements **encryption** at the communications level. The de facto standards for communication based encryption are the **Secure Sockets Layer** (**SSL**) and **Transport Layer Security** (**TLS**) protocols. The infrastructure used to support communication-based encryption is **PKI** which is composed of a number of cryptographic technologies but provides for two key services, data integrity and confidentiality. **Public Key** systems involve a **Certificate Authority** (**CA**) responsible for issuing a pair of digital **certificates**: one public and one private. The public key, as its name suggests, may be freely disseminated. This key does not need to be kept confidential. The **Private Key**, on the other hand, must be kept secret. The owner of the key pair must guard the private key closely, as sender authenticity and non-repudiation are based on the signer having sole access to the private key. There are several important characteristics of these key pairs. First, while they are mathematically related to each other, it is impossible to calculate one key from the other. Therefore, the private key cannot be compromised through knowledge of the associated public key. Second, each key in the key pair performs the inverse function of the other. What one key does, only the other can undo.

The CA is a trusted third party that issues digital certificates to its subscribers, binding their identities to the key pairs they use to sign electronic communications digitally. Digital certificates contain the name of the subscriber, the subscriber's public key, the digital signature of the issuing CA, the issuing CA's public key, and other pertinent information about the subscriber and the subscriber's organization. The CA can revoke certificates upon private key compromise, separation from an organization, etc. These certificates are stored in an on-line, publicly accessible repository. The repository, referred to as **Certificate Revocation List** (**CRL**), also maintains an up-to-date listing of all revoked but not yet expired certificates.

For the DoD PKI, users interface with the ***Real Time Automated Personnel Identification System (RAPIDS)*** workstation via the ***Issuance Portal*** for digital certificates residing on the **Common Access Card** (**CAC**).

To guarantee that data stays confidential and secure from attackers listening on the network in promiscuous mode (i.e., network sniffers) and to provide better performance, **Symmetric Encryption** (secret key) is used to encrypt and decrypt the data. **Asymmetric Encryption** (public key-private key) is not used for all encryption because it is too expensive for high volume data. For SSL and TLS, Asymmetric Encryption is used initially to pass the **secret key** (often called the **session key**). Once the secret key has been established on both sides, all subsequent data communications can be performed using Symmetric Encryption.

There are at least two options when an application needs to support PKI/SSL: use a DoD-approved **module** or develop the application abiding by the **DoD Class 3 Public Key Infrastructure Interface Specification**. The guidance linked to this perspective applies to **Public Key Enabled** applications wanting to operate within the DoD PKI.

Guidance

- G1308: Configure **Public Key Enabled** applications to use a **Federal Information Processing Standard** (**FIPS**) 140-2 certified cryptographic module.
- G1309: Make applications handling high value unclassified information in Minimally Protected environments **Public Key Enabled** to interoperate with **DoD High Assurance**.
- G1310: Protect application cryptographic objects and functions from tampering.
- G1311: Use **Hypertext Transfer Protocol over Secure Sockets Layer** (**HTTPS**) when applications communicate with DoD **Public Key Infrastructure** (**PKI**) components.
- G1312: Make applications capable of being configured for use with DoD **PKI**.
- G1313: Provide documentation for application configuration for use with DoD **PKI**.

Part 5: Developer Guidance > Software Security > Technologies and Standards for Implementing Software Security > Key Management

P1041: Key Management

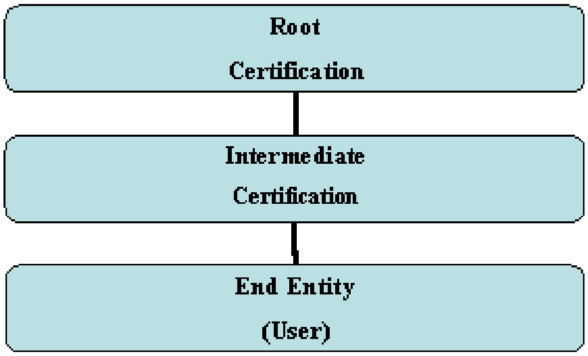
The key enabler in the **PKE** applications is **Asymmetric Encryption**, the use of **public** and **private keys**. It is used in exchanging **session keys**, and it is used to verify **Certificates**; therefore, it is critical for applications to manage and protect the keys used in **PKI**. This includes the associated technologies used to store the keys and Certificates. The following list of guidance addresses key management issues.

Guidance

- G1314: Provide applications the ability to import **Public Key Infrastructure** (**PKI**) software certificates.
- G1316: Ensure that applications protect **private keys**.
- G1317: Ensure applications store **Certificates** for subscribers (the owner of the **Public Key** contained in the Certificate) when used in the context of signed and/or encrypted email.
- G1318: Develop applications such that they provide the capability to manage and store **trust points** (**Certificate Authority Public Key Certificates**).
- G1319: Ensure applications can recover data encrypted with legacy keys provided by the DoD **PKI Key Recovery Manager** (**KRM**).
- G1942: Provide applications the ability to export **Public Key Infrastructure** (**PKI**) software certificates.

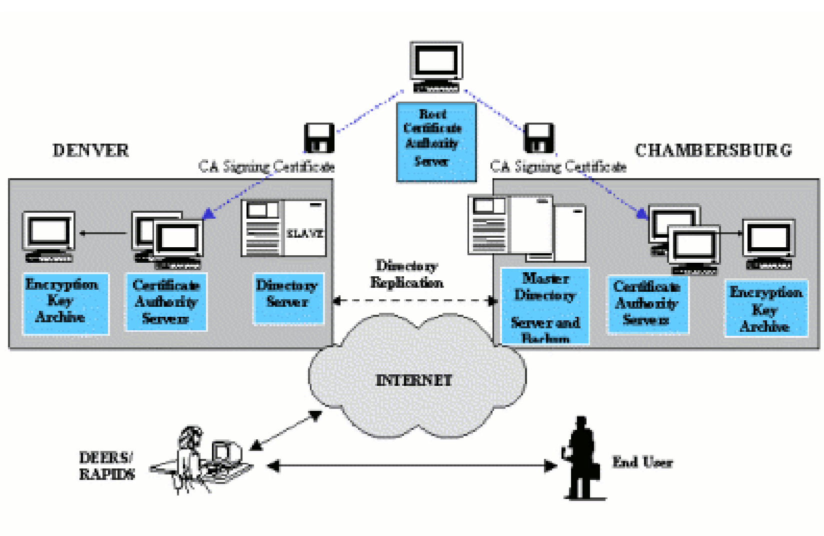
P1009: Certificate Processing

The **DoD** implementation of the **Public Key Infrastructure** (**PKI**) is the framework and services that provide for the generation, distribution, control, tracking and destruction of **Public Key Certificates**. The purpose of a PKI is to manage keys and **Certificates** in a way whereby the DoD can maintain a trustworthy networking environment. Digital Certificates are issued by a DoD **Certificate Authority**. It is an electronic document that contains a user's **identity**, a pubic key, a validity period, and the issuing authority. It is digitally signed and the Certificate is chained hierarchically in a path that can be traced to the Root Certificate.



I1091

Certificates can be sent via email or more commonly retrieved from repositories (**Directory Server**). Applications must validate the Certificate by checking status of the Certificate. There are two forms of status checking, the legacy **Certificate Revocation List** (**CRL**) or **Online Certificate Status Protocol** (**OCSP**). The status check determines whether a Certificate is revoked. A Certificate can be revoked if the information in the Certificate may have changed (relocation, new email) or the Certificate has been compromised. The Certificate validation is a critical part of the PKI process; it is the application's responsibility to perform the status checks. The following guidance sets the guidelines for the Certificate processing.



I1093

Guidance

- G1327: Enable an application to obtain new **Certificates** for subscribers.
- G1328: Enable an application to retrieve **Certificates** for use, including relying party operations.

- G1330: Ensure applications are capable of checking the status of **Certificates** using a **Certificate Revocation List** (**CRL**) if not able to use the **Online Certificate Status Protocol** (**OCSP**).
- G1331: Ensure applications are able to check the status of a Certificate using the **Online Certificate Status Protocol** (**OCSP**).
- G1333: Only use a **Certificate** during the Certificate's validity range, as bounded by the Certificate's "Validity - Not Before" and "Validity - Not After" date fields.
- G1335: Make applications capable of being configured to operate only with PKI Certificate Authorities specifically approved by the application's owner/managing entity.
- G1338: Ensure that **Public Key Enabled** applications support multiple organizational units.

P1315: Smart Card Logon

Smart Card Logon (SCL), also called Cryptographic Logon (CLO), capability enables users to log onto their unclassified network using their **Common Access Card** (**CAC**) and associated Personal Identification Number (PIN) instead of a username and password.

This capability addresses the Department of Defense (DoD) mandate in DoD Instruction 8520.2 [R1206] to Public Key (PK) enable all unclassified networks for certificate-based authentication to DoD information systems. SCL provides the increased security of two-factor authentication by allowing users to access their network with something they have (their CAC with DoD issued certificates) and something they know (their PIN).

> ***Note:*** *Joint Task Force-Global Network Operations (JTF-GNO) Communications Tasking Orders (CTOs; for example, CTO 06-02 and CTO 07-015) provide specific implementation directions for DoD, to include non-Windows-based operating systems (see https://www.cybercom.mil/default.aspx; DoD PKI required). Additional Mobile Code policy information is available from the* ***Information Assurance Support Environment*** *Web site, http://iase.disa.mil/mcp/index.html; DoD PKI required.*

Before enabling SCL, each unclassified network must also meet the following requirements:

- Implement **Active Directory** in the root domain
- Equip user workstations with a DoD-approved Windows operating systems, smart card readers, drivers, and the appropriate version of middleware
- Populate Active Directory accounts with each user's **Electronic Data Interchange Personal Identifier** (**EDI-PI**) numbers associated with the CAC certificates

Once users start using SCL to access their unclassified networks, they no longer need to remember their ever-changing and complex network passwords. SCL is a more secure method of network logon because the PIN is not stored on or transmitted over the network.

The following process illustrates how to use the PKI certificate for network logon:

- The user inserts the user's CAC into the smart card reader attached to the workstation, and, when prompted, enters the user's CAC PIN instead of a username and password
- A secure process retrieves the PKI certificate from the CAC and verifies it is valid and from a trusted issuer
- The user's workstation verifies the network domain controller's certificate is valid and from a trusted issuer
- If the user's PKI certificate and the domain controller certificate are valid, the user is automatically logged onto the network

> ***Note:*** *There are certain user groups (e.g., system administrators) that are unable to use PKI Certificates on a CAC as the primary token for smart card logon. A DoD CIO memo of 14 August 2006, Approval of the Alternate Logon Token (available via Defense Knowledge Online, https://www.us.army.mil/ [user account and DoD PKI Certificate required] DoD PKE Knowledge Base Library Smart Card and Alternate Token folders) permits the use of an Alternate Logon process.*

The Defense Manpower Data Center (DMDC) Common Access Card site (http://www.dmdc.osd.mil/smartcard) contains additional information, reports and developer support concerning the DoD CAC implementation.

Guidance

- G1862: Configure **Active Directory** for **Smart Card** Logon.
- G1869: Configure Domain Controllers for **Smart Card** Logon.

P1387: XML Digital Signatures

XML signatures are a form of **digital signatures** applied to digital content including XML; XML signatures are represented as XML, but the signed data may be any collection of digital content. XML signatures are usually used to sign XML documents or portions thereof. XML signatures as defined in NESI, particularly in this perspective, are specified by the W3C recommendation *XML Signature Syntax and Processing*.

XML signatures often serve as electronic versions of signatures. XML signatures provide a means to implement non-repudiation and detect changes to signed content.

Signing XML content is more complicated than signing other digital content, since XML has more than one syntactically correct way to express data. Because digital signatures are based on a hash of the signed content, a singe byte difference in the signed content can cause a verification of the digital signature to fail. The following examples show ways to represent different syntactically correct XML documents that may be semantically equivalent in a given context.

- White space is often insignificant within XML documents (<Node > is syntactically identical to <Node>).
- Order of XML attributes may vary.
- Nodes within an XML document may have different XPath representations (for example using a relative path versus an absolute path).
- Namespace prefixes may have different name but represent to same namespace.
- Namespaces declarations may occur in any order.
- XML Element attributes may vary in order.
- Child elements may inherent namespaces from parent elements which creates portability issues for signed nodes that are moved from one XML document to another.
- Line break characters may vary between operating systems.
- Order of XML nodes can vary or be unspecified.
- XML comments may vary between XML documents.

Because XML allows these different representations within XML documents, it is necessary to conduct a **canonicalization** of the XML document before signing a XML document and before verifying a signature of an XML document. Unfortunately existing canonicalization specifications are insufficient in some case and impact the interoperability and use of XML digital signatures. In some cases, it is necessary for developers to conduct their own canonicalization of XML as a precondition before signing the XML and again before verifying the signature of the signed XML to ensure consistency between the signed and verified documents and to account for inconsistencies for which the current canonicalization specification do not account.

In addition to issues relating to canonicalization and signature creation and verification, there is a potential to abuse digital signatures to conduct denial of service, cross-site scripting, or replay attacks through the use of carefully crafted XSLT and XPath expressions. To work around these issues, developers often employ a number of best practices to limit or reduce the impacts of such attacks. The W3C is drafting a collection of such best practices for the practical and secure use of XML digital signatures: http://www.w3.org/TR/xmldsig-bestpractices/. In addition to these best practices, NESI provides a number of guidance and best practice statements for the use of XML digital signatures.

The following links provide additional information for XML Digital Signatures and Canonicalization specifications.

- W3C Recommendation, *XML Signature Syntax and Processing (Second Edition)*, 10 June 2008, http://www.w3.org/TR/xmldsig-core/
- W3C Recommendation, *Canonical XML Version 1.1*, 2 May 2008, http://www.w3.org/TR/2008/REC-xml-c14n11-20080502/
- W3C Recommendation, *Exclusive XML Canonicalization Version 1.0*, 18 July 2002, http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/

Guidance

- G1366: Digitally sign all **messages** where non-repudiation is required.

- G1367: Digitally sign **message** fragments that are required not to change during transport.
- G1371: Use the **National Institiure of Standards and Technology** (**NIST**) ***Digital Signature Standard*** promulgated in the **Federal Information Processing Standards** Publication 186 (**FIPS** Pub 186-3 as of June 2009) for creating **Digital Signatures**.
- G1902: Use the Exclusive Canonicalization algorithm when digitally signing **XML** content that may be embedded in another XML document.

Best Practices

- BP1903: Include an xsd:dateTime field within long-lived **XML** digital signatures.

P1020: Encryption Services

Successful implementation of **Public Key** enabled applications is predicated on the correct selection and use of security algorithms. This section provides guidance on the use of **encryption**, **digital signature**, and authentication services in a consistent manner to interoperate with DoD **PKI**.

Guidance

- G1320: Use a minimum of 128 bits for **symmetric keys**.
- G1321: Enable applications to be capable of performing **Public Key** operations necessary to verify signatures on DoD **PKI** signed objects.
- G1322: Ensure that applications that interact with the DoD **PKI** using **SSL** (i.e., **HTTPS**) are capable of performing cryptologic operations using the **Triple Data Encryption Algorithm** (**TDEA**).
- G1323: Generate random **symmetric encryption** keys when using symmetric encryption.
- G1324: Protect **symmetric keys** for the life of their use.
- G1325: Encrypt **symmetric keys** when not in use.
- G1326: Ensure applications are capable of producing **Secure Hash Algorithm** (**SHA**) **digests** of **messages** to support verification of DoD **PKI** signed objects.
- G1797: Use a minimum of 1024 bits for **asymmetric keys**.

P1085: SOAP Security

Several security challenges arise from implementing a typical **service-oriented architecture** using **SOAP** including the following:

- **Authentication** (ensure that the sender of the **message** is genuine)
  - Preventing **identity** spoofing when accessing to a Web service.
  - Preventing tampering with the **WSDL** file of a Web service provider in order to spoof an endpoint.
- **Integrity** (ensure that an unauthorized third party cannot change a message during transmission without detection)
  - Preventing the interception of a message to or from a Web service provider to change its contents.
- **Confidentiality** (ensure that a message cannot be read by an unauthorized third party during transmission)
  - Preventing the interception of a message to or from a Web service provider and to obtain privileged information.

These security challenges are commonly addressed at the communication layer, the message layer, or both. The **Secure Sockets Layer** (**SSL**) and **Transport Layer Security** (**TLS**) protocols are commonly applied to the communication layer to provide confidentiality and authentication (both one-way and two-way authentication of service producers and consumers); see the Authorization and Access Control [P1339] perspective for further information.

Industry standards organizations such as the **World Wide Web Consortium** (**W3C**) and **Organization for the Advancement of Structured Information Standards** (**OASIS**) address these threats at the message level by specifying standards for providing authentication, protecting integrity and ensuring confidentiality. A common set of message layer specifications in the SOAP security space includes the following:

- Web Services Security (WS-Security) provides message layer mechanisms for implementing SOAP security. WS-Security supports message integrity through the use of XML Digital Signatures, support message confidentiality through the use of XML Encryption, and support authentication through the use of credentials such as X.509 certificates, **Security Assertion Markup Language** (**SAML**) tokens, and username/passwords.
- XML Digital Signatures provide a means to implement non-repudiation and detect changes to signed content. See the XML Digital Signatures [P1387] perspective for additional information.
- XML Encryption provides confidentiality by specifying a process for encrypting data (arbitrary data to include XML content). The result of the encryption processes is an XML element containing or referencing the encrypted data. XML Encryption can be selectively applied to data (for example to only parts of a XML document).
- SAML specifies ways to exchange security information (such as authentication, authorization, and attribute information related to assertions) across security domains. See the Security Assertion Markup Language [P1189] perspective for more information.
- **eXtensible Access Control Markup Language** (**XACML**) is a specification used in conjunction with SAML to represent and exchange access control policies across an enterprise.
- Web Services Policy (WS-Policy) describes a model and syntax for Web services to describe its requirements (required security policies, supported encryption algorithms, message delivery reliability requirements, etc.).
- WS-Trust specifies ways to issue, renew, obtain, and validate security tokens used to create trust relationships between participants in a secure message exchange.

Guidance

- G1357: Do not rely solely on transport level security like **SSL** or **TLS**.
- G1359: Bind **SOAP Web service** security policy assertions to the service by expressing them in the associated **WSDL** file.
- G1362: Validate XML messages against a **schema**.
- G1363: Do not use clear text passwords.
- G1364: Hash all passwords using the combination of a timestamp, a **nonce** and the password for each **message** transmission.

- G1365: Specify an expiration value for all security tokens.
- G1366: Digitally sign all **messages** where non-repudiation is required.
- G1367: Digitally sign **message** fragments that are required not to change during transport.
- G1369: Digitally sign all requests made to a security token service.
- G1371: Use the **National Institute of Standards and Technology** (**NIST**) ***Digital Signature Standard*** promulgated in the **Federal Information Processing Standards** Publication 186 (**FIPS** Pub 186-3 as of June 2009) for creating **Digital Signatures**.
- G1372: Use an X.509 **Certificate** to pass a **Public Key**.
- G1373: **Encrypt messages** that cross an **IA** boundary.
- G1374: Individually **encrypt** sensitive **message** fragments intended for different intermediaries.
- G1376: Do not **encrypt** message fragments that are required for correct **SOAP** processing.

Best Practices

- BP1360: Use the **XML** Infoset standard to serialize messages.
- BP1375: Use **asymmetric encryption** for sensitive **SOAP**-based **Web services**.

P1189: Security Assertion Markup Language (SAML)

The **Security Assertion Markup Language** (**SAML**) is a vendor-neutral protocol specification for software applications and services to exchange security information in a distributed network environment. The SAML specification, maintained by the **OASIS** Security Services Technical Committee, defines schemas for how security assertions are structured and embedded within transport protocols.
SAML defines three types of assertions for an individual or machine:

Authentication	used for proving identity
Authorization	used for controlling access
Attributes	used to provide additional details to constrain the request

Email address, employee number, and rank are examples of attribute assertions.
SAML does not define any implementation of the services that authenticate or authorize users. Commercial vendors provide implementations in the form of authentication servers to authenticate and authorize users. Authentication servers respond to SAML requests and return SAML assertions that ensure the subject is logged in and authorized to access the resource.

Guidance

- G1379: Use **SAML** version 2.0 for representing security assertions.
- G1380: Use the **XACML** 2.0 standard for **SAML**-based rule engines.

P1064: RDBMS Security

Relational Database Management Systems remain on top amidst emerging technologies such as **XML** and **Object-Oriented Database Management Systems**. The continued dominance of **relational databases** is unlikely to change in the near future. First, there is still a large amount of legacy data and legacy applications that rely on **RDBMS**. Second, RDBMS are continuing to evolve to integrate XML as a function of the database. RDBMS is a reliable and proven technology that will be here for the long run. This perspective provides guidance on how best to secure the database.

Guidance

- G1346: Audit database access.
- G1347: Secure remote connections to a database.
- G1348: Log database **transactions**.
- G1349: Validate all input that will be part of any dynamically generated **SQL**.
- G1350: Implement a strong password policy for **RDBMS**.
- G1351: Enhance database security by using multiple user accounts with constraints.
- G1352: Use database clustering and redundant array of independent disks (RAID) for high availability of data.

Best Practices

- BP1353: Use a data abstraction layer between the RDBMS and application for externally-visible applications to prevent the disclosure of sensitive data.
- BP1355: Do not design the database around the requirements of an application.

P1042: LDAP Security

The **Lightweight Directory Access Protocol** (**LDAP**) can be thought of as a datastore. It is an open Internet standard produced by the **Internet Engineering Task Force** (**IETF**). LDAP is, like X.500, both an information model and a protocol for querying and manipulating it. The LDAP overall data and namespace model is essentially that of X.500. The major difference is that the LDAP protocol itself is designed to run directly over the **TCP/IP** stack, and it lacks some of the more esoteric DAP protocol functions. LDAP can store text, photos, **URLs**, pointers to whatever, binary data, and Public Key **Certificates**.

Guidance

- G1377: Use **LDAP** 3.0 or later to perform all connections to LDAP repositories.
- G1378: Encrypt communication with **LDAP** repositories.

P1039: JNDI Security

The **Java Naming and Directory Interface** (**JNDI**) is an **API** for directory services in a **Java EE** environment. It allows **clients** to discover and look up data and objects using a name. JNDI is portable and independent of the actual implementation. Additionally, it specifies a **service provider** interface (SPI) that allows plugging **directory service** implementations into the framework. The JNDI service implementations are hidden from the user and may make use of a **server**, a flat file, or a database. The choice is up to the JNDI provider.

Guidance

- G1071: Use vendor-neutral interface connections to the enterprise (e.g., **LDAP**, **JNDI**, **JMS**, databases).
- G1079: Use **deployment descriptors** to isolate configuration data for **Java EE** applications.
- G1239: Use **design patterns** (e.g., **facade**, **proxy**, or **adapter**) or property files to isolate vendor-specifics of vendor-dependent connections to the enterprise.

Best Practices

- BP1116: If using **Java**-based messaging (e.g., **JMS**), register destinations in **Java Naming and Directory Interface** (**JNDI**) so **message clients** can use JNDI to look up these destinations.

Examples

```
// Step 1
// Create a hashtable that contains the parameters
// used to initialize JNDI.
Hashtable contextParams = new Hashtable();
// Step 2
// Specify the context factory to use. The context
// factory is provided by the
// implementation.
contextParams.put( Context.INITIAL_CONTEXT_FACTORY, "com.jnidprovider.ContextFactory");
// Step 3
// The next parameter is the URL specifying the location
// of the JNDI provider's data store
contextParams.put( Context.PROVIDER_URL, "http://jndiprovider-database");
// Step 4
// Create the JNDI provider's context.
Context navyCurrentContext= new InitialContext ( contextParams );
// Step 5
// Look up the desired bean using its full name.
Object reference= navyCurrentContext.lookup ( "mil.us.navy.NavyBean" );
// Step 6
// Cast the located bean to the desired type.
MyBean navyBean= (NavyBean) PortableRemoteObject.narrow ( reference );
```

P1005: Application Resource Security

Applications use and store a large amount of data that often do not go into databases. For instance, an application often uses configuration files for application configuration, preferences files for personalization information (custom user experience) and resource files for internationalization support. Apply appropriate protection to sensitive resources to prevent attackers from tampering. Application bundles, properties files, configuration files when tampered could cause the user to execute inappropriate commands, expose sensitive data due to invalid configuration or cause the application to be inoperable. Therefore, it is of utmost importance to take appropriate measures to protect these resources.

Guidance

- G1344: Encrypt sensitive data stored in configuration or resource files.

P1038: Java Security

Java is an **Object Oriented Language**; applications benefit from the encapsulation features which offers protection for application data. Java was also designed and built with security in mind. Some of the security features include restricting direct access to memory (protecting data access privileges), array bounds checking (buffer overflow), and ability to install a security manager to protect resources. Despite all the security features built into the Java language, it does not mean that Java **APIs** are immune to security problems. Take care in the design and implementation of APIs to prevent attacks. The following security guidance are targeted to Java-specific APIs.

Guidance

- G1341: Use a security manager support to restrict application access to privileged resources.
- G1342: Restrict direct access to class internal variables to functions or methods of the class itself.
- G1343: Declare classes final to stop inheritance and prevent methods from being overridden.

Part 5: Developer Guidance > Software Security > Policies and Processes for Implementing Software Security

P1392: Policies and Processes for Implementing Software Security

Many software errors and exploits share similar root causes resulting from the failure to follow common high level best practices. The detailed perspectives listed below provide best practices to enable compliance with policies and processes for implementing software security.

The Secure Coding and Implementation Practices [P1316] perspective provides a high level overview of important areas for consideration during software development from a programming language independent viewpoint. It discusses software security activities and best practices for use throughout the development lifecycle.

Protecting Data at Rest has become increasingly critical given Information Technology trends toward utilizing highly mobile computing devices and removable storage media. The Data at Rest [P1360] perspective provides guidance for complying with the DoD memorandum *Encryption of Sensitive Unclassified Data at Rest on Mobile Computing Devices and Removable Storage Media* [R1330] which mandates encryption not only for **Personally Identifiable Information** (**PII**) information but for all non-publicly released unclassified information contained on mobile computing devices and removable storage media.

The Mobile Code [P1314] perspective provides guidance to comply with DoD Instruction 8552.01, *Use of Mobile Code Technologies in DoD Information Systems* [R1292]. This Instruction identifies DoD-defined mobile code risk categories, describes their characteristics, and establishes restrictions for the acquisition (to include development) and use of mobile code technologies assigned to each risk category. This instruction applies to all DoD-owned or DoD-controlled information systems used to process, transmit, store, or display DoD information including mobile devices.

Detailed Perspectives

- Secure Coding and Implementation Practices [P1316]
- Data at Rest [P1360]
- Mobile Code [P1314]

Best Practices

- BP1868: Incorporate mechanisms to enhance Computing Infrastructure (CI) availability.

P1316: Secure Coding and Implementation Practices

Many software errors and exploits share similar root causes resulting from the failure to follow common high level best practices. This perspective provides insight into a few of the major secure coding and implementation best practices from a programming language independent viewpoint.

This perspective does not provide all required guidance and best practices for secure software development. However, it does strive to provide a high level overview of important areas for consideration during software development. Finally, this perspective serves as a resource for additional information and tools for building secure software.

For best effectiveness, software security activities should occur throughout the development lifecycle. For example, security requirements (such as required roles, privacy requirements, accreditation requirements, etc.) are captured during the requirement phase of software system development. During the design phase, high level concepts such as defense in depth and principal of least privilege are applied. During actual development, programmers follow predefined development practices to include applying a coding standard. Finally, unit testing, regression testing, and peer reviews test the developed software for security vulnerabilities and policies.

Detailed Perspectives

- Apply Principle of Least Privilege [P1317]
- Practice Defense in Depth [P1318]
- Apply Secure Coding Standards [P1319]
- Apply Quality Assurance to Software Development [P1320]
- Validate Input [P1321]
- Heed Compiler Warnings [P1322]
- Handle Exceptions [P1323]

P1317: Apply Principle of Least Privilege

To minimize risk and side effects due to possible security vulnerabilities, each process, function, or method within a software system should execute with the minimal set of privileges necessary to complete the action. To enable execution of code with the minimal set of privileges required, separate code requiring access to different resources or higher privileges. Whenever it is necessary to have an elevated permission level to complete an action, the elevated permission should be held for a minimum time. This approach reduces the chance that a security exploit can execute arbitrary code and minimizes the impact when an exploit occurs.

Best Practices

- BP1881: Separate code based on required privilege.
- BP1889: Minimize execution at elevated privilege levels to the shortest time required.

P1318: Practice Defense in Depth

A good practice to manage risk is to have multiple layers of defensive strategies. This reduces risk, since an exploit in one layer of defense may be stopped by another layer of defense and therefore eliminate or limit the consequences of the exploit.

As an example, a software system may use **Secure Sockets Layer** (**SSL**), **Public Key Infrastructure** (**PKI**), WS-Security along with **SOAP**, and provide security in integrity using database stored procedures, triggers and views.

Guidance

- G1301: Practice layered security.

Best Practices

- BP1922: Design systems to have security as a core capability.

Part 5: Developer Guidance > Software Security > Policies and Processes for Implementing Software Security > Secure Coding and Implementation Practices > Apply Secure Coding Standards

P1319: Apply Secure Coding Standards

Develop to a documented coding standard for each target development language and platform to minimize the likelihood of security vulnerabilities caused by programmer error. This coding standard should include secure coding practices but may also include standards and policies that improve readability or maintainability.

Guidance

- G1215: Provide a coding standards document.

P1320: Apply Quality Assurance to Software Development

Quality assurance techniques are a useful tool in identifying and eliminating security vulnerabilities. Source code audits and peer reviews should be a regular activity during software development and maintenance along with normal testing activities.

To the extent possible, utilize automated tools to assist in verifying that code meets standards as defined in the applicable coding standard document. This will result a more repeatable process and shorten the time required for a peer reviews.

Guidance

- G1304: Unit test all code.

Best Practices

- BP1944: Peer review source code.

P1321: Validate Input

Proper input validation can eliminate many software vulnerabilities. Do not limit validation to the presentation tier; rather, all implementations of external facing modules should validate inputs prior to use. This can help prevent attacks including SQL Injection, Cross-Site Scripting, Buffer Overflows, and Denial of Service.

Validation may include checking lengths of input parameters to prevent buffer overflows. It may also include checking input against a list of allowed or disallowed characters to prevent execution of arbitrary code.

Guidance

- G1032: Validate all input fields.
- G1147: Use **domain analysis** to define the constraints on input data validation.
- G1302: Validate all inputs.
- G1339: Practice defensive programming by checking all method arguments.
- G1349: Validate all input that will be part of any dynamically generated **SQL**.
- G1362: Validate XML messages against a **schema**.

P1322: Heed Compiler Warnings

Many run time errors are detectable during the compilation process. Compiler warnings are often useful in detecting possible violations of syntax rules and mistakes introduced by developers which may lead to run time errors. For example, a compiler may warn about use of the assignment operator "**`=`**" instead of the equality operator "**`==`**" inside an **`if`** statement or warn about unchecked buffer assignment which could lead to a buffer overflow resulting in the execution of arbitrary code.

A good security practice to prevent many of these errors is to detect them at compile time by compiling code using the highest warning level available for the compiler. Compilers often have a warning option which enables additional warnings, for instance the GCC **`-Wall`** flag and the Java **`-Xlint`** option. In many cases, these options only enable the most common warnings and additional flags are required. Detailed understanding of the specific warning capabilities of a given compiler are necessary to ensure that all of the desired warnings truly are enabled.

Upon receiving an error from the compilation process, developers should modify the code to remove the deficiency or explicitly document the code stating the reason the code is valid but still produces a warning. Some programming languages and compilers contain syntax for documenting such exception to compiler warnings and suppressing the warning from the compiler output.

> ***Note:*** *Compiler warnings may vary depending on the compiler used and the target platform.*

Best Practices

- BP1890: Compile code using the highest compiler warning level available.
- BP1891: Develop code such that it compiles without compiler warnings.
- BP1892: Explicitly document exceptions for valid code that produces compiler warnings.

P1323: Handle Exceptions

Exception objects can convey sensitive information through their message or exception type. Translate information from exceptions to display meaningful information to users without displaying sensitive information from the exception. For example, do not expose the file layout of a system to a user through an exception thrown during file access. When necessary, catch and sanitize internal exceptions before re-propagating them to other parts of the system or displaying the exception to the user.

Guidance

- G1094: Catch all exceptions for application code exposed as a **Web service**.
- G1340: Log all exceptional conditions.

Best Practices

- BP1893: Return meaningful, but non-sensitive, information from exception handlers.

P1360: Data at Rest

Protecting **Data at Rest** (**DAR**) has become increasingly critical given Information Technology trends toward utilizing highly mobile computing devices and removable storage media. **Personally Identifiable Information** (**PII**) or sensitive government information stored on devices such as laptops, thumb drives and personal digital assistants (PDAs) is often unaccounted for and unprotected. This can pose a problem if the devices containing PII are compromised, lost, or stolen. This has generated negative media attention and potentially exposed sensitive information.

DAR technologies allow protection of data stored on mobile computing devices in the event of theft or other loss by way of encryption and password protection, thus enhancing **information assurance** (**IA**) posture. **DoD**, concerned not only with the loss of PII but with all unclassified data contained on mobile devices, issued a memorandum on 3 July 2007 entitled *Encryption of Sensitive Unclassified Data at Rest on Mobile Computing Devices and Removable Storage Media*.[R1330] This memo mandates encryption not only for PII records, but for all non-publicly released unclassified information contained on mobile computing devices and removable storage media. The cryptography used in the DAR technologies must be **National Institute of Standards and Technology** (**NIST**) **Federal Information Processing Standard** (**FIPS**) 140-2 compliant.

The DoD memo also mandates that all new computer assets procured to support the DoD enterprise include a **Trusted Platform Module** (**TPM**) version 1.2 or higher where such technology is available. TPM is a microcontroller that stores keys, passwords and digital certificates. It typically is affixed to the motherboard of computers. The nature of this hardware chip ensures that the information stored becomes more secure from external software attack and physical theft.

A U.S. General Services Agency (GSA) announcement on 14 June 2007 [R1334] notified **Chief Information Officers** (**CIOs**) that SmartBUY awarded Government-wide contractual agreements in May 2007 for DAR encryption commercial solutions to protect sensitive data. The GSA announcement identified contract awardees and provided a list of DAR encryption products available through the DoD SmartBUY Enterprise Software Initiative (ESI).

Guidance

- G1381: Encrypt sensitive persistent data.
- G1895: Encrypt all Unclassified DoD **Data at Rest** (**DAR**) not releasable to the public stored on mobile computing devices.
- G1896: Use **Data at Rest** (**DAR**) products that are **Federal Information Processing Standard** (**FIPS**) 140-2 compliant.
- G1897: Purchase **Data at Rest** (**DAR**) encryption products that are included in the Enterprise Software Initiative (ESI).

Best Practices

- BP1898: Purchase computers which contain a **Trusted Platform Module** (**TPM**).

P1314: Mobile Code

Mobile code is software obtained from remote systems, transferred across a network, and then downloaded and executed on a local system without explicit installation or execution by the recipient.

Conventional executable code refers to typical program code or software that is not embedded in data or text and that the user knowingly executes. Conventional executable code includes both compiled and interpreted code; examples include compiled C or Ada programs, scripts written in JavaScript or VBScript, Java applications, and binary .exe files.

Mobile code and ***active content*** are not interchangeable terms; incorrect usage can result in confusion. Mobile code is a broad term encompassing code obtained from a remote system that downloads across a network and executes on a local machine without the user's explicit initiation or knowledge. Active content is the term used to describe executable code embedded within (or bound to) text or data that executes automatically without explicit user initiation. Examples of active content include Microsoft Visual Basic for Applications (VBA) macros embedded in Microsoft Word and Excel files, PostScript commands embedded in PostScript documents, and scripts embedded in Macromedia Director and Shockwave movies.

As depicted in the figure below, mobile code is comprised of that active content or conventional executable code which has become "mobile." When active content and/or conventional executable code resides statically on the workstation or host on which it executes, it is not mobile code. However, when such code originates from an external system, traverses a network, downloads onto a workstation or host, and executes without explicit user initiation, it becomes mobile code.

I1218: Mobile Code

Mobile code brings many benefits to a computer system, such as reduction of communication, ability to perform asynchronous tasks, dynamic software deployment, and temporary and scalable applications. But despite all the benefits there are many threats that mobile agents bring to a computer system, such as denial of service, destruction, unauthorized access, breach of privacy, and theft of resources, among others. These threats are related to protection of the host systems and mobile code systems themselves.

The Department of Defense issued DoD Instruction 8552.01, *Use of Mobile Code Technologies in DoD Information Systems* [R1292], in October 2006 to establish and implement DoD mobile code policy. This Instruction identifies DoD-defined mobile code risk categories, describes their characteristics, and establishes restrictions for the acquisition (to include development) and use of mobile code technologies assigned to each risk category. It also establishes restrictions on the use of mobile code in email and emerging mobile code technologies and directs monitoring to detect the presence of prohibited mobile code. Any prohibited mobile code discovered must be removed.

This instruction applies to all DoD-owned or DoD-controlled information systems used to process, transmit, store, or display DoD information. This includes mobile devices (e.g., cellular phones, handheld devices) capable of executing mobile code. Mobile code that originates from and travels exclusively within a single enclave boundary is exempt from the requirements of DoD Instruction 8552.01. However, if an enclave consists of geographically dispersed computing environments that are connected by the **Unclassified but Sensitive Internet Protocol Router Network** (**NIPRNet**), **Secret Internet Protocol Router Network** (**SIPRNet**), **Internet**, or a public network, the requirements of this instruction apply.

Category 1 Mobile Code

Category 1 mobile code technologies exhibit a broad functionality, allowing unmediated access to workstation, server, and remote system services and resources. Category 1 mobile code technologies have known security vulnerabilities with few or no countermeasures once they begin executing. Execution of Category 1 mobile code typically requires an all-or none decision: either execute with full access to all system resources or do not execute at all.

The following mobile code technologies are assigned to ***Category 1A*** (allowed):

- ActiveX controls
- Shockwave movies (including Xtras)

The following mobile code technologies are assigned to ***Category 1X*** (prohibited):

- Mobile code scripts that execute in Windows Scripting Host (WSH) (e.g., JavaScript and VBScript downloaded via a Uniform Resource Locator (URL) file reference or email attachment)
- HTML Applications (e.g., `.HTA` files) that download as mobile code
- Scrap objects
- Microsoft Disk Operating System (MS-DOS) batch scripts
- Unix shell scripts
- Binary executables (e.g., `.exe` files) that download as mobile code

The use of unsigned Category 1 mobile code in DoD information systems is prohibited.

Category 2 Mobile Code

Category 2 mobile code technologies have full functionality, allowing mediated or controlled access to workstation, server, and remote system services and resources. Category 2 mobile code technologies may have known security vulnerabilities but also have known fine-grained, periodic, or continuous countermeasures or safeguards.

The following mobile code technologies are currently assigned to ***Category 2***:

- Java applets
- Visual Basic for Applications (i.e., Visual Basic for Applications [VBA] macros)
- PostScript
- Mobile code executing in the Microsoft .NET Common Language Runtime
- PerfectScript
- LotusScript

Category 2 mobile code that does not execute in a constrained execution environment may be used in DoD information systems if the mobile code is obtained from a trusted source over an assured channel. Information regarding these assured channels is available from DoD Instruction 8552.01.

Category 3 Mobile Code

Category 3 mobile code technologies support limited functionality, with no capability for unmediated access to workstation, server, and remote system services and resources. Category 3 mobile code technologies may have a history of known vulnerabilities, but also support fine-grained, periodic, or continuous security safeguards.

The following mobile code technologies are currently assigned to ***Category 3***:

- JavaScript, including Jscript and ECMAScript variants, when executing in the browser
- VBScript, when executing in the browser
- Portable Document Format (PDF)
- Flash

Category 3 mobile code technologies may be freely used without restrictions in DoD information systems.

Emerging Mobile Code Technologies

Emerging mobile code technologies refer to all mobile code technologies, systems, platforms, or languages whose capabilities and threat level have not yet undergone a risk assessment and been assigned to one of the three risk categories described above.

Some examples of emerging technologies follow:

- Microsoft's .NET Framework, when used to execute mobile code
- The flat script files used by Java WebStart to control the execution of Java applications

Because of the uncertain risk, the use of emerging mobile code technologies in DoD information systems is prohibited.

Mobile Code in Email

Mobile code can be embedded in an email body or an email attachment and can be downloaded as part of the actual email. Alternately, mobile code residing on a remote server can be referenced from within an email body or attachment and can be automatically downloaded and executed. Some types of mobile code execute automatically as soon as the user clicks on the message subject or previews the message; others execute when the user opens an attachment containing mobile code. Email viruses, worms, and Trojan horses typically utilize mobile code technologies; they are forms of malicious mobile code sent to users via email.

Due to the significant risk of malicious mobile code downloading into user workstations via email, and the ease of rapidly spreading malicious mobile code via email, the following restrictions apply to all types of mobile code in email independent of risk category:

- To the extent possible, the automatic execution of all categories of mobile code in email bodies and attachments is disabled, compliant with DoD mobile code policy implementation guidance.
- To the extent possible, mobile code-enabled software is configured to prompt the user prior to opening email attachments that may contain mobile code.

Code-Signing Certificate Requirements

DoD code-signing certificates (i.e., their associated private keys) are used to sign Category 1A mobile code that will reside on DoD-owned or DoD-controlled servers prior to its installation on the servers. When code signing is used to meet the requirements for Category 2 mobile code that will reside on DoD-owned or DoD-controlled servers, the mobile code is signed with DoD code-signing certificates prior to its installation on the servers. DoD code-signing certificates are designated as trusted by default by all Components. DoD-owned and DoD-controlled servers are trusted sources by default.

Guidance

- G1883: Use a DoD PKI code signing certificate to sign mobile code residing on DoD-owned or DoD-controlled servers.
- G1884: Configure browsers to use Category 1A allowed mobile code per DoD Instruction 8552.01. [R1292]
- G1885: Configure browsers to disable Category 1X prohibited mobile code per DoD Instruction 8552.01. [R1292]
- G1886: Disable automatic execution of mobile code in email clients.
- G1887: Monitor configured mobile code-enabled software to ensure it is in compliance with DoD Instruction 8552.01. [R1292]

Best Practices

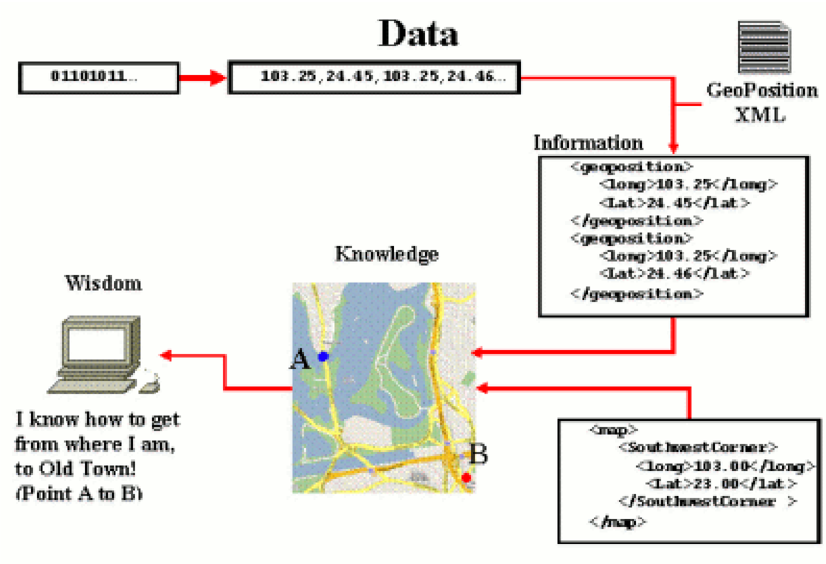
- BP1888: Only enable plaintext viewing in email clients on DoD-owned and DoD-operated information systems.

P1012: Data

There are several common definitions of data; the NESI Glossary definition includes the following points:

- **Data** is unprocessed **information**.
- Data is information without context.

But both of these definitions rely on the term "information" which can be a circular definition back to data. To clarify this, the following model helps create definitions of Information, **Knowledge** and **Wisdom**. Data flows into the **system** as a set of zeros and ones. The system transforms this initial data into other data that is more understandable from a human perspective (i.e., a list of double precision, floating point numbers). If the numbers are placed into a context such as it is a geographic position, then the data starts to become Information. As information is combined together, the result is referred to as Knowledge (i.e., the knowledge of where one is). When the knowledge can support making decisions, the results are Wisdom (i.e., how to get from point A to point B).



I1112

Within NESI, the term Data covers the entire data spectrum (i.e., Information, Knowledge and Wisdom) with a focus on the transfer of data between components. NESI helps Program Managers understand and implement DoD governing directives for net-centricity and interoperability to include the **DoD Net-Centric Data Strategy** (**NCDS**).

Generic data guidance statements include guidelines relative to basic functions associated with the definition of data and the most general categories of data types. Examples of the most basic data functions include **data modeling** and **domain analysis**. The most general categories of data types include **relational database** data and **XML**.

Data Exposure defines the steps necessary to set up the **metadata** infrastructure associated with a net-centric data strategy. This infrastructure permits the exposure (i.e., visibility) of net-centric data to the user community. This infrastructure will be set up once but maintained to include the following:

- Registry where the metadata will reside
- Repository where the data will reside
- Rules applicable to the tagging of data

Tagging and metadata rules follow from Data Categorization. Generic Data Categorization includes data types that adhere to **XML Schema** rules. Specialty Data Categories, such as **Electronic Data Interchange** (**EDI**) and **Binary XML** include data types that do not fit in the current XML paradigm but for which special XML extensions may be developed.

Data Publishing defines the steps necessary to make data available within the net-centric data strategy infrastructure. It requires the project to have a **Community of Interest** (**COI**), a model of the data associated with the project and an **ontology** which taken together can be used as a basis for structural metadata. Based on the Data Categorization rules promulgated in the data exposure section appropriate tags are determined and applied to the data.

There are many ways to persist data to include storing data on a **file system** or in a database (e.g., **hierarchical databases**, **object-oriented databases**, **native XML databases**, and **relational databases**). For more detailed information regarding data within a **Node**, see the Node Data Strategy [P1329] perspective.

Detailed Perspectives

- XML [P1083]
- Metadata Registry [P1050]
- Data Modeling [P1003]
- Metadata [P1049]
- Relational Database Management Systems [P1063]

P1083: XML

The **Extensible Markup Language** (**XML**) is a **World Wide Web Consortium** (**W3C**) initiative that allows encoding **data** and information with meaningful structure and **semantics** into a document that computers and humans can read easily. XML is ideal for information exchange and is easily extended to include other data types. The ubiquitous nature of XML within existing and proposed DoD projects has spawned a lot of activity to capture guidelines and requirements that facilitate net-centricity and interoperability. Many of these activities have not been finalized and are "emerging" from a NESI viewpoint. This NESI Perspective leverages the work done by Roger Costello and colleagues at xFront.com. It is by no means complete, but it does provide a starting point for additional DoD XML work.

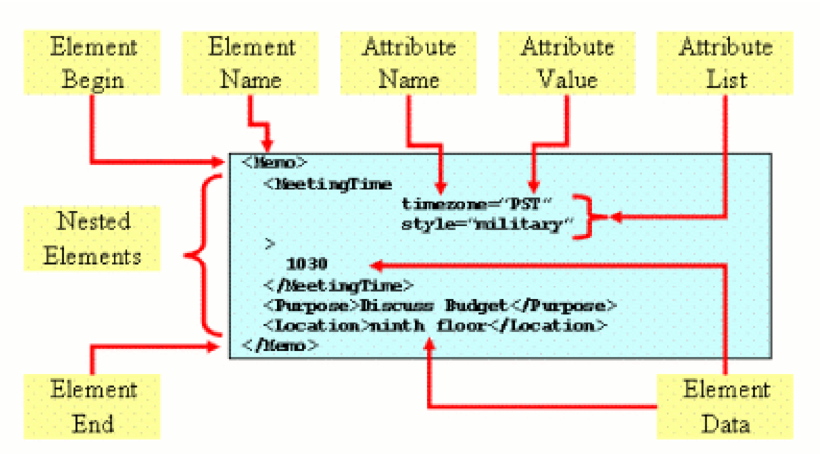
There are two key measures of XML instance document correctness: being **well-formed** and **valid**. Those concepts and others are introduced in the perspectives in the following subsection list.

Detailed Perspectives

- XML Syntax [P1095]
- XML Semantics [P1096]
- XML Processing [P1105]

P1095: XML Syntax

The syntax of an **XML document** is a hierarchical collection of **XML elements** that identify the name of the **data** within the XML document and the value associated with the element. Elements can have **attributes** and be nested within other elements. The following is a simplistic XML document displayed in **ASCII** with the major syntactical **components** labeled.



I1173

Guidance

- G1724: Develop **XML documents** to be **well formed**.

Best Practices

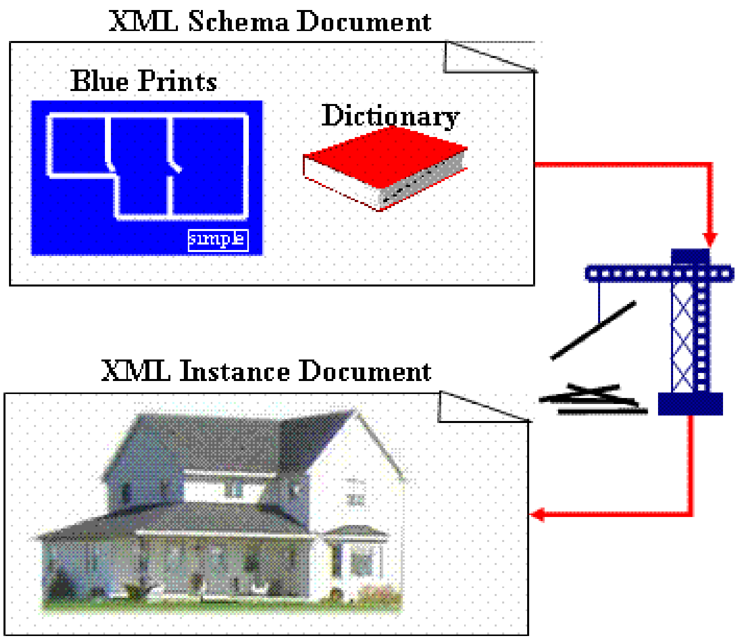
- BP1258: Explicitly define the encoding style of all data transferred via **XML**.
- BP1752: Place dynamic **XML element** data within an XML CDATA section.

Examples

An example of an XML instance document is the following weather information XML. It can be thought of as a complex data structure that contains a weather station's data.

P1096: XML Semantics

The semantics of an **XML document** are limited to the structural composition of data, the relationships of the structures to each other, and the rules governing data content. A full semantic interpretation of the **XML** content must be left to humans or tools that humans have written that connote some meaning to the data. For example, the semantics captured by XML might define a weather station that is comprised of air temperature, soil temperature, anemometer and hygrometer and the values and units associated with these values. XML does not capture what this data means semantically to a pilot or soldier.



I1174

The semantics of any XML instance document are captured in another XML document called the schema which is also defined using XML; see the two perspectives in the following subsection list.

Detailed Perspectives

- XML Schema Documents [P1097]
- XML Instance Documents [P1104]

P1097: XML Schema Documents

An **XML Schema** is a **W3C** specification for defining the **semantics** and structure of **XML documents**. For a discussion of the grammar that governs **XML** see the XML Syntax [P1095] perspective. The semantics are limited to the structural composition of data, the relationships of the structures to each other, and the rules governing data content. More detailed discussions of the schema documents are in the related perspectives in the following subsection list.

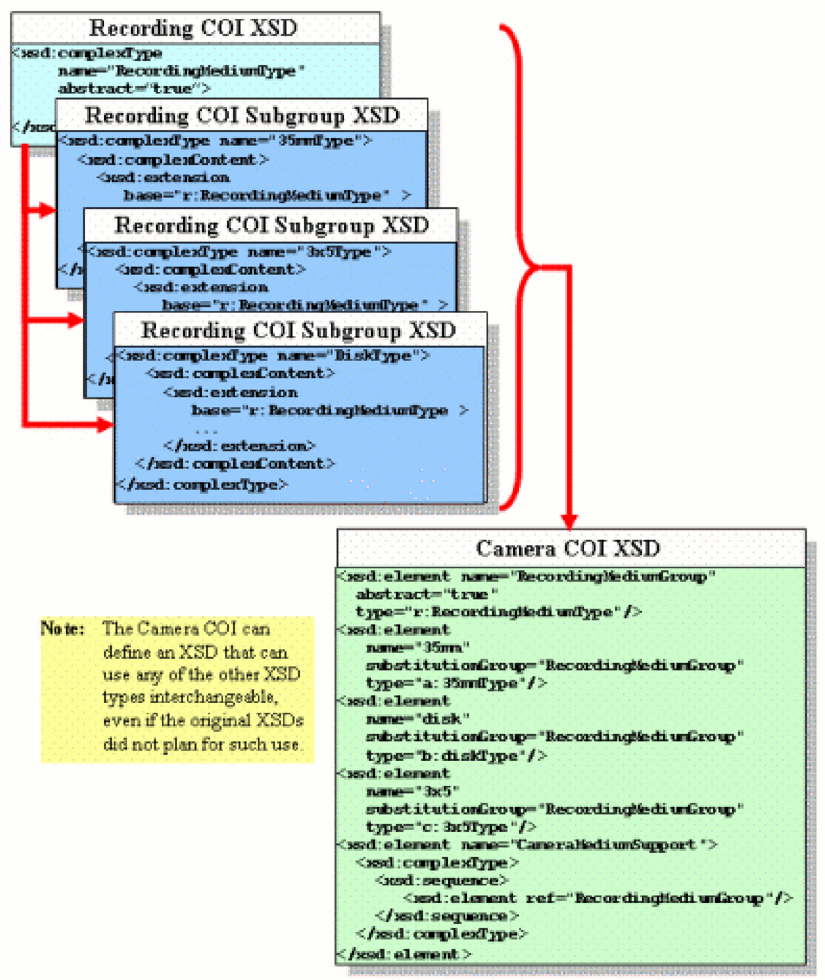
Detailed Perspectives

- Using XML Substitution Groups [P1102]
- Defining XML Types [P1101]
- XML Schema Files [P1099]
- Using XML Namespaces [P1100]
- Defining XML Schemas [P1098]
- Versioning XML Schemas [P1103]

P1102: Using XML Substitution Groups

Substitution groups allow using elements defined in externally defined and controlled schemas as interchangeable elements in new schemas. More specifically, elements can be assigned to a special group of elements that are said to be substitutable for a particular named element called the head element. Elements in a substitution group must have the same type as the head element, or they can have a type that has been derived from the head element's type. See the *XML Schema Part 0: Primer Second Edition* at http://www.w3.org/TR/xmlschema-0/#SubsGroups for further information.

Substitution groups allow any of the element members' substitution group elements to participate as a member of a more abstract concept. For example, in the following **XML**, **`RecordingMedium`** is the name of the substitution group. The members of the group are the **`RecordingMedium`** element itself and **`35mm`**, **`disk`** and **`3x5`**. Anywhere that **`RecordingMedium`** is used as a reference, **`35mm`**, **`disk`** and **`3x5`** can also be used. For a complete example study the following diagram that defines a **`CameraMediumSupport`** element that has a single sequence comprised of the **`RecordingMediumGroup`** substitution group.



I1175

Guidance

- G1731: Only reference **XML elements** defined by a Type in substitution groups.
- G1744: Only reference abstract **XML elements** in substitution groups.
- G1745: Append the suffix Group to substitution group **XML element** names.

P1101: Defining XML Types

The **W3C** defined datatype as follows:

"A datatype is a 3-tuple, consisting of a) a set of distinct values, called its value space, b) a set of lexical representations, called its lexical space, and c) a set of facets that characterize properties of the value space, individual values or lexical items."
[See W3C "XML Schema Part 2: Datatypes Second Edition," Section 2.1, http://www.w3.org/TR/xmlschema-2/#typesystem]
There are two kinds of datatypes definable within XML: Primitive and Derived. Primitive datatypes are not defined in terms of other datatypes while Derived datatypes are defined in terms of other datatypes. All datatypes can be further classified as Built-in and User-derived. Built-in datatypes are those which have been defined by the W3C in XML Schema Part 2: Datatypes Second Edition. User-derived datatypes are those defined by individual schema designers.

The guidance included in this perspective is for primitive and derived datatypes designed by individual schema designers.

Guidance

- G1727: Provide names for XML type definitions.
- G1728: Define types for all **XML elements**.
- G1729: Annotate XML type definitions.
- G1740: Append the suffix Type to XML type names.

Best Practices

- BP1732: Follow the **Upper Camel Case** (**UCC**) naming convention for XML Type names.

P1099: XML Schema Files

Schema definitions are usually captured in files. The following guidance applies to those files which actually contain the schema definitions.

Guidance

- G1735: Use the **`.xsd`** file extension for files that contain XML Schema definitions.
- G1736: Separate document schema definition and document instance into separate documents.

Examples

```
<?xml version="1.0"?>
<xsd:schema xmlns: xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.camera.org"
            xmlns: nikon="http://www.nikon.com"
            xmlns: olympus="http://www.olympus.com"
            xmlns: pentax="http://www.pentax.com"
            elementFormDefault="unqualified">
 <xsd:import namespace="http://www.nikon.com"/>
 <xsd:import namespace="http://www.olympus.com"/>
 <xsd:import namespace="http://www.pentax.com"/>
 <xsd:element name="Camera">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="body"
                     type="nikon:BodyType"/>
       <xsd:element name="lens"
                     type="olympus:LensType"/>
       <xsd:element name="ManualAdapter"
                     type="pentax:manual_adapter_type"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

P1100: Using XML Namespaces

A **namespace** defines the scope for schema components and de-conflicts the use of schema components. Qualifying prefixes simplify the use of namespaces in names by appending a qualifier onto the beginning of the name that is mapped to a particular schema. Namespaces can become quite confusing if they are not used consistently.

Guidance

- G1085: Establish a **registered namespace** in the **XML Gallery** in the **DoD Metadata Registry** for all DoD Programs.
- G1383: Use a **registered namespace** in the XML Gallery in the **DoD Metadata Registry**.
- G1384: Review **XML Information Resources** in the **DoD Metadata Registry**, using those which can be reused.
- G1385: Identify **XML Information Resources** for registration in the XML Gallery of the **DoD Metadata Registry**.
- G1737: Define a target namespace in schemas.
- G1738: Define a qualified namespace for the target namespace.

Best Practices

- BP1739: Use the xsd qualifying prefix for XML Schema namespace.
- BP1741: Do not provide a schema location in import statements in schemas.
- BP1742: Use the xsi qualifying prefix for XML Schema instance namespace uses.

P1098: Defining XML Schemas

While it is possible to use **Document Type Definitions** (**DTD**) to convey much of the same information as the **XML Schema Definition** (**XSD**), XSDs have several distinct advantages which are very useful in terms of interoperability. **XML Schemas** have richer support for defining and using types than DTDs which capture domain information such as allowable ranges and units. For example, XSDs can define an elevation type with values limited to meters in the range of 0 to 12,000.

Guidance

- G1045: Separate **XML** data presentation **metadata** from data values.
- G1725: Develop XML documents to be **valid** XML.
- G1726: Define XML Schemas using **XML Schema Definition** (**XSD**).
- G1730: Follow a documented **XML** coding standard for defining **schemas**.

Best Practices

- BP1732: Follow the **Upper Camel Case** (**UCC**) naming convention for XML Type names.
- BP1733: Follow the **Upper Camel Case** (**UCC**) naming convention for **XML element** names.
- BP1734: Follow the **Lower Camel Case** (**LCC**) naming convention for **XML attributes**.

P1103: Versioning XML Schemas

XML Schemas capture the **semantics** of the **data** that the schemas define. As the understanding of the data and its interrelationships evolves, the need to redefine the semantics captured by the schema is inevitable. This evolution can have a wide ranging ripple effect throughout a large widely distributed system or family of systems. Therefore, the uniform managing of schema versions is essential.

Guidance

- G1004: Make public **interfaces** backward-compatible within the constraints of a published **deprecation** policy.
- G1019: Deprecate public interfaces in accordance with a published deprecation policy.
- G1727: Provide names for XML type definitions.
- G1753: Declare the XML schema version with an **XML attribute** in the root **XML element** of the schema definition.
- G1754: Give each new XML schema version a unique **URL**.

P1104: XML Instance Documents

An **XML instance document** is an **XML document** which is defined by an **XML Schema** but is populated with the actual data whereas the schema is the definition of the structure and semantics of data (**metadata**).

Guidance

- G1725: Develop XML documents to be **valid** XML.
- G1736: Separate document schema definition and document instance into separate documents.

Best Practices

- BP1742: Use the xsi qualifying prefix for XML Schema instance namespace uses.
- BP1743: Use .xml as the file extension for files that contain XML Instance Documents.

P1105: XML Processing

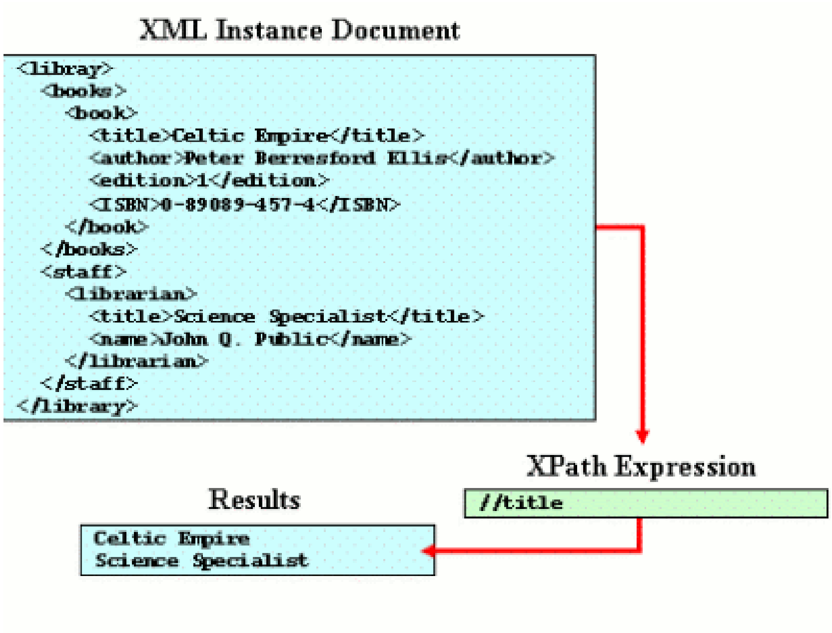
One of the primary benefits of using **XML** is that it can be read by humans or processed by software. The perspectives in the following subsection list pertain to XML processing.

Detailed Perspectives

- XPath [P1107]
- XSLT [P1106]
- Parsing XML [P1109]
- XML Validation [P1110]

P1107: XPath

A **valid XML Document** is a representation of a **Document Object Model** (**DOM**) tree structure. Each of the XML elements is considered a node with the tree. **XML Path Language** (**XPath**) is a succinct and elegant way of addressing the individual nodes (i.e., elements) within the tree (i.e., document) or to perform basic computations on the Element Data within the document. The following is a very simplistic example of how an XML Document and XPath work together. The XML instance document contains the data and the XPath provides the instructions on how to traverse the document.



I1172

For a more detailed description of XPath, see the following W3C location: http://www.w3.org/TR/xpath; there also is an XPath tutorial at http://www.w3schools.com/xpath/default.asp.

Guidance

- G1756: Isolate XPath expression statements into the configuration data.

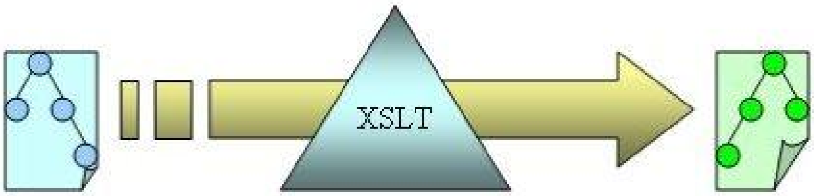
Best Practices

- BP1757: Do not ignore namespace prefixes in XPath expressions.
- BP1758: Make names in descendant expressions unique within an XML document.

Part 5: Developer Guidance > Data > XML > XML Processing > XSLT

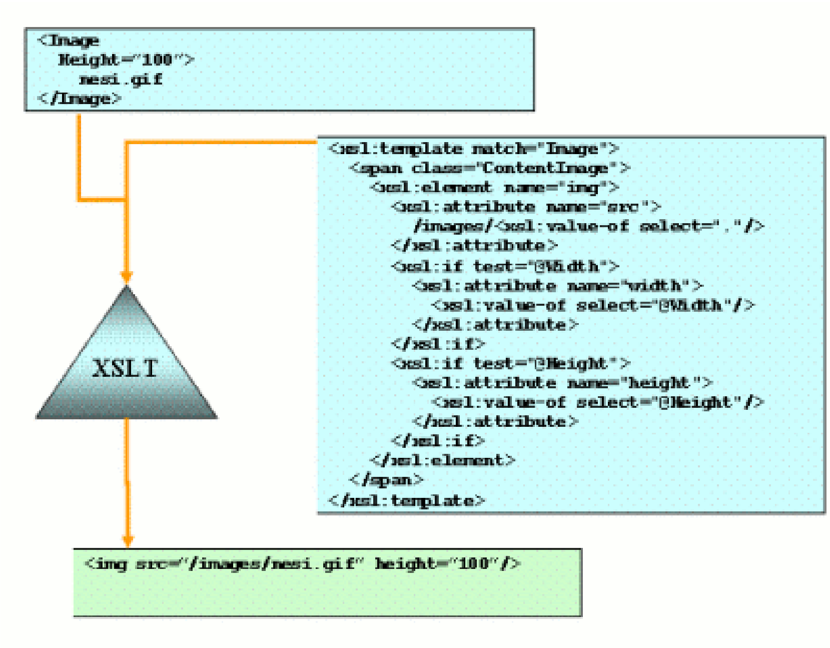
P1106: XSLT

XSL Transformations (**XSLT**) allow **XML** data transformation using the functional **eXtensible Stylesheet Language** (**XSL**).



I1176

XSL is dependent on **XML Path Language** (**XPath**) to address nodes within the input document. For XPath guidance and best practices see the XPath [P1107] perspective. The following example produces **HTML** image tag from an image **XML element** with optional height and width attributes.



I1177

Templates

Use templates to transform particular sections of an XML document tree. XSLT requires at least one template which matches to an absolute path of an element (e.g., `/`). Inside of a template, match other templates by using **`xsl:apply-templates`**. Passing an XPath query to the select parameter of **`xsl:apply-templates`** constructs a list of nodes by which templates are compared and executed.

XSLT 2.0

XSLT 2.0 improves on XSLT 1.0 and adds functionality that was previously only achieved through proprietary language extensions.

Some of the more significant improvements include the following:

- Backwards-compatibility
- Improved XPath functions
- Regular expressions

- Schema validation to temporal and result trees
- Multiple outputs
- Aggregation
- Strong data typing

Guidance

- G1746: Develop XSLT **style sheets** that are XSLT version agnostic.
- G1751: Document all XSLT code.
- G1755: Use accepted file extensions for all files that contain XSL code.

Best Practices

- BP1747: Use the xsl qualifying prefix for XSLT namespace.
- BP1748: Separate static content from transformational logic in XSLTs.
- BP1749: Use xsl:include for including XSL transforms.
- BP1750: Use xsl:import for reusing XSL code.

P1109: Parsing XML

One advantage of **XML** is that a variety of standard **parsers** are available to parse documents. Another advantage is that the consumer of the XML document is free to choose the type of parser to use.

A couple of common types of XML parsers include the **Document Object Model** (**DOM**) and Simple API for XML (SAX) parsers. The DOM parser uses a tree-based approach, while the SAX parsers use an event-based approach. Both approaches have advantages and disadvantages depending the application.

In addition to the various types of XML parsers, there are multiple implementations of each types of parser. This provides the developer great flexibility in choosing an XML parser implementation. To take advantage of this flexibility, the developer must take care when developing software to allow for changing the XML parser throughout the life-cycle of the software. One way to do this is to provide a wrapper or adapter class that isolates the XML parser implementation allowing for changes to the XML parser during development or deployment.

Best Practices

- BP1769: Provide wrapper or adapter classes to isolate XML parser implementations.

Part 5: Developer Guidance > Data > XML > XML Processing > XML Validation

P1110: XML Validation

One advantage of **XML** is that it allows for validation of **XML instance documents**. Validation can occur at the producer and/or consumer or anywhere in-between.

Guidance

- G1725: Develop XML documents to be **valid** XML.

Best Practices

- BP1265: Validate **XML** documents during document generation.

P1050: Metadata Registry

A Metadata Registry is a central repository for storing and maintaining **metadata** definitions. A metadata registry typically has the following characteristics:

- It is a protected area where only approved individuals may make changes
- It stores **data elements** that include both semantics and representations
- The semantic areas of a metadata registry contain the meaning of a **Data Element** with precise definitions
- The representational areas define how the data is represented in a specific format such as within a database or a structure file format such as **XML**

Metadata registries often are stored in an international format called **ISO-11179**.

A metadata registry is frequently set up and administered by an organization's **data architect** or data modeling team.

The **DoD Metadata Registry** provides a common source of data information required to promote interoperability in the Net-Centric Data Environment.

In the Net-Centric Data Strategy, data sources are called **Data Assets** which are divided into two generic areas:

The **data** area includes the following:

- XML stored in repositories (files)
- **Database data**
- Data services
- Data streams (real time)
- Sensor data
- **Message** data (includes **EDI**)

The metadata area includes the following:

- Metadata stored in registries
  - **UDDI**
  - Electronic Business Using eXtensible Markup Language (ebXML)
  - DoD Metadata Registry
  - Other **ISO/IEC 11179 Registries**
  - Discovery metadata stored in Catalogs
- DoD Discovery Metadata Standard (**DDMS**)
- Interface Metadata (**WSDL**)
- Structural Metadata (**XSD**)

Data comes in many forms. It can be simple or complex; structured or unstructured in nature.

Simple Structured Data has an uncomplicated **data structure** . All requisite metadata is provided and simple data types only are used (e.g., integers, long integers, strings, and simple lists).

Simple Unstructured Data has uncomplicated data structure but not all requisite metadata is provided.

Complex Structured Data has well-defined metadata. It includes data represented in **XML documents** with deeply hierarchical and recursive structures. Complex data can be represented in a complex data structure or can be mapped into a relational or flat structure with additional metadata provided to represent the complex relationships. Although complex structured data is generically a property of object oriented databases, the Complex Data Structures can be filled from any source.

- Data
    - XML files
    - defined by **XML Schemas** (**XSDs**)
        - **Interface**
- Metadata stored in DoD Repository
    - XML Schemas (XSDs)
    - Discovery metadata
        - **WSDL**
        - **UDDI**
    - Web Service Source Code
    - XSDs include element validation and descriptions
    - XSDs may import other XSDs
    - XSDs are validated
    - Complex Structured Data follows all of the XML rules.

Complex Semi-Structured Data has partial metadata. It includes data defined in **COBOL** copybooks and Electronic Data Interchange standards **ANSI X.12** and Health Level 7 (HL7). Semi-structured data can be as complex or more so as any Complex Structured data. It can map into or be XML. It may also be missing some Metadata or an XSD.

Complex Unstructured Data has little or no metadata. It includes data in binary files, spreadsheets, documents, and print streams.

Guidance

- G1125: Use the **Department of Defense Metadata Specification** (**DDMS**) for standardized tags and taxonomies.
- G1141: Base **data models** on existing data models developed by **Communities of Interest** (**COI**).
- G1382: Be associated with one or more **Communities of Interest** (**COIs**).
- G1383: Use a **registered namespace** in the XML Gallery in the **DoD Metadata Registry**.
- G1384: Review **XML Information Resources** in the **DoD Metadata Registry**, using those which can be reused.
- G1385: Identify **XML Information Resources** for registration in the XML Gallery of the **DoD Metadata Registry**.
- G1386: Review predefined commonly used **data elements** in the **Data Element Gallery** of the **DoD Metadata Registry**, using those in the **relational database** technology which can be reused in the Program.
- G1387: Identify **data elements** created during Program development for registering in the **Data Element Gallery** of the **DoD Metadata Registry**.
- G1388: Use predefined commonly used database tables in the **DoD Metadata Registry**.
- G1389: Publish database tables which are of common interest by registering them in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.
- G1391: Identify **taxonomy** additions or changes in conjunction with the **Communities of Interest** (**COIs**) during the Program development for potential inclusion in the **Taxonomy Gallery** of the **DoD Metadata Registry**.

Best Practices

- BP1392: Register services in accordance with a documented service registration plan.
- BP1855: Identify types of data items for potential sharing external to the program.
- BP1856: Identify specific data items for potential sharing external to the program.
- BP1857: Prioritize data items for potential sharing external to the program.
- BP1858: Publish preliminary program data-related development plans.
- BP1859: Create external representations for sharable data items.

- BP1860: Create **metadata** representations for sharable data items.
- BP1861: Publish data access services that implement interfaces to shared data.
- BP1863: Make shareable data assets visible, even if they are not accessible.

P1003: Data Modeling

Modeling is an essential step in understanding the data that will comprise a system. Before implementing a system, it is important to understand the basic **data elements** and the relationships of the elements. The end products of **data modeling** can be **XML schemas**, **RDBMS** schema definitions or the data portion of objects.

Rather than conducting data modeling efforts in isolation, seek out and identify relevant **communities of interest** (**COIs**). Doing so will provide for more effective data models that build upon lessons learned, provide lessons learned to the greater community, reduce costs through reuse, and enhance interoperability through the use of common semantics across the community. One way to do this is to base new data models on the terminology published by relevant COIs listed in the **Taxonomy Gallery** of the **DoD Metadata Registry**. Another is to look for relevant COIs outside of the DoD. Examples of common high level COI data models follow.

Universal Core (UCore)

UCore is a federal information sharing initiative that supports the *National Strategy for Information Sharing* (available at http://georgewbush-whitehouse.archives.gov/nsc/infosharing/index.html) and associated Departmental and Agency strategies. UCore enables information sharing by defining an implementable specification (XML Schema) containing agreed upon representations for the most commonly shared and universally understood concepts of who, what, when, and where.

UCore is designed to be simple to understand, explain, and implement. It is small, containing a minimal set of objects with broad applicability across a wide range of domains. UCore is built on an extensible framework that permits users to build more detailed exchanges tailored to their mission or business requirements. UCore is based on and leverages existing commercial and governmental standards. The UCore validation processes and tools provide a means to achieve consistently definable levels of interoperability, promoting machine understanding between both anticipated and unanticipated users.

For more information on UCore, including developer guides, tutorials, examples, and validation tools, see the Universal Core 2.0 site: http://metadata.ces.mil/ucore/ (user registration required).

National Information Exchange Model (NIEM)

The NIEM represents a partnership of the U.S. Departments of Justice and Homeland Security. It is designed to develop, disseminate and support enterprise-wide information exchange standards and processes that can enable jurisdictions to share critical information effectively in emergency situations, as well as support the day-to-day operations of agencies throughout the nation. NIEM objectives include the following:

- Bring stakeholders and communities of interest together to identify information sharing requirements in day-to-day operational and emergency situations
- Develop standards, a common lexicon and an on-line repository of information exchange package documents to support information sharing
- Provide technical tools to support development, discovery, dissemination and reuse of exchange documents
- Provide training, technical assistance and implementation support services for enterprise-wide information exchange

For more documentation, training, and tools to support the NIEM, see the NIEM site: http://www.niem.gov.

Cursor on Target (CoT)

CoT is a data strategy for enabling DoD systems to exchange much needed time sensitive position or ***what***, ***when*** and ***where*** information. The CoT data strategy is based on a terse CoT XML Schema and a set of sub-schema extensions. The CoT schema is available on the DoD Metadata Registry [R1227]. Further CoT information is available at http://cot.mitre.org (user registration required).

Joint Consultation, Command and Control Information Exchange Data Model (JC3IEDM)

JC3IEDM is a data model developed by the Multilateral Interoperability Programme (MIP) Data Modeling Working Group. The aim of the MIP is to achieve international interoperability of Command and Control Information Systems (C2IS) at all levels. The MIP cooperates to develop a data modes that describe the information that allied component commanders need to exchange (both vertically and horizontally) and serve as the common interface specification for the exchange of essential battlespace information. The JC3IEDM is evolving from the Command and Control Information Exchange Data Mode (C2IEDM) data modeling efforts. Both data models are available on the MIP site.[R1070]

Common Alerting Protocol (CAP)

CAP is a simple but general format for exchanging all-hazard emergency alerts and public warnings over all kinds of networks. CAP is developed and managed by **Organization for the Advancement of Structured Information Standards** (**OASIS**). CAP allows a simultaneous dissemination of consistent warning message over many different warning systems, thus increasing warning effectiveness while simplifying the warning task. CAP facilitates the detection of emerging patterns in local warnings of various kinds, such as might indicate an undetected hazard or hostile act, and CAP provides a template for effective warning messages based on best practices identified in academic research and real-world experience. The current version of the Common Alerting Protocol is available at http://www.oasis-open.org/specs/.

Naval Architecture Elements Reference Guide (NAERG)

NAERG is a key component of the coordinated set of activities intended to create a Department of the Navy (DON) Enterprise Architecture (EA). The NAERG supports the consistent and aligned development of architecture products across the DON, by implementing a common and reusable lexicon for naming the various elements within the federated DON EA. Further information see the NAERG site: https://sadie.spawar.navy.mil/Wiki/NAERG (DoD PKI Certificate required).

Guidance

- G1141: Base **data models** on existing data models developed by **Communities of Interest** (**COI**).
- G1144: Develop two-level database models: one level captures the **conceptual** or logical aspects, and the other level captures the **physical** aspects.
- G1147: Use **domain analysis** to define the constraints on input data validation.
- G1148: **Normalize** data models.
- G1382: Be associated with one or more **Communities of Interest** (**COIs**).
- G1384: Review **XML Information Resources** in the **DoD Metadata Registry**, using those which can be reused.
- G1386: Review predefined commonly used **data elements** in the **Data Element Gallery** of the **DoD Metadata Registry**, using those in the **relational database** technology which can be reused in the Program.
- G1388: Use predefined commonly used database tables in the **DoD Metadata Registry**.
- G1391: Identify **taxonomy** additions or changes in conjunction with the **Communities of Interest** (**COIs**) during the Program development for potential inclusion in the **Taxonomy Gallery** of the **DoD Metadata Registry**.

Best Practices

- BP1145: Use vendor-neutral **conceptual/logical models**.
- BP1254: For **command-and-control** systems, use the names defined in the Joint Command, Control and Consultation Information Exchange Data Model (JC3IEDM) for data exposed to the outside communities.
- BP1394: Identify, publish and validate data objects exposed to the enterprise early in the data engineering process and update in a spiral fashion as development proceeds.
- BP1396: Develop high-level conceptual data models for new systems prior to Milestone A based on the business process context in which the system will be used.
- BP1397: Identify and develop use cases or reuse existing use cases as appropriate as early in the data engineering process as possible to support **data model** development.
- BP1398: Develop Interaction models as appropriate.
- BP1400: Programs will use authoritative **metadata** established by the Joint Mission Threads (JMTs) when available.

- BP1857: Prioritize data items for potential sharing external to the program.
- BP1858: Publish preliminary program data-related development plans.
- BP1859: Create external representations for sharable data items.
- BP1860: Create **metadata** representations for sharable data items.
- BP1901: Use Universal Core (UCore) as the basis for information exchange models for systems that exchange internal data with external systems.

P1049: Metadata

Services and **data** to be mediated should always be formally defined, and typically this is done with some form of computer readable **metadata**.

NESI currently requires metadata, defined primarily as **XML Schema** and **Web Services Description Language** (**WSDL**) documents, be registered in the **DoD Metadata Registry**. NESI further specifies rules system developers must follow in developing XML Schema, including the requirement to search the registry for existing schemas that can be reused, aligning new schemas as closely as possible to existing similar schemas, reviewing schemas with the DoD XML Namespace Manager, and looking for other relevant Government and industry schemas that could be leveraged. The purpose is to avoid unnecessary duplication of effort and improve the success of future interoperability through common definitions.

The NCES Data Strategy team, including the maintainers of the DoD Metadata Registry, strives to create a common data model, per **Community of Interest** (**COI**); but recognizing the difficulty in accomplishing that goal the team promotes the use of "mediation" from one schema to another. NCES currently implements mediation simply through the use of eXtensible Style Language Transformations (**XSLT**) to transform **XML documents** from one schema to another.

This focus on centrally managed data models is not viable as a long term solution to mediation since it requires substantial effort to define accurate transformations, and the underlying "business objects" almost always lose information in the process. The vision of a non-redundant object model is considered by most experts as unachievable due to social and communications barriers among the hundreds of organizations working as part of or with the Federal Government and the DoD in particular.

Accepting the fact that use of the DoD Metadata Registry is a requirement gives rise to posing the question should there be a new **FORCEnet** COI "**namespace**," or should the FORCEnet activities simply try to find suitable existing namespaces in which to register their metadata. Clearly, some FORCEnet applications will be able to leverage some of the existing schemas. But are there a significant number of new schemas to be registered, and if so can they be aligned to existing COI namespaces or will there be unacceptable barriers to introducing the changes required.

Moreover, the technologies for application and system development continue to improve to allow more rapid turnaround of new software capabilities, and in fact software developers are finding less of a need to work at the XML document level at all. **Model Driven Architecture** (**MDA**) technology, for example, is becoming mainstream, and **interfaces** are being developed visually, with the schemas automatically generated according to the graphical model. The creation of interfaces and schemas is becoming more of a dynamic activity, and the projected ad hoc interoperability of loosely coupled components, enforced by the FORCEnet vision, will mean bureaucratic processes such as those introduced by the DoD Metadata Registry may introduce significant risk.

Striving to minimize the number of schema variations by leveraging common schemas across applications is laudable and should be encouraged. However, more advanced solutions to mediation are critical to the interoperability problem where common schemas do not exist. This may require a more dynamic process for registering metadata, without restrictions. An argument can be made for a FORCEnet COI in this regard.

As promoted by the NCES Data Strategy team, XSLT is the common practice for mediation. However, XSLT only solves a single point-to-point integration, and it is limited in its ability to support semantic validation. The Web Services Business Process Execution Language (WS-BPEL) [R1347] is an **OASIS** standard for defining specific interactions among services using documents defined through schema. It can use XSLT and other technologies to perform transformation of data elements, and semantics are implicit through their use. However, each BPEL definition is limited even further to a single **use-case** for the data.

Reduce the work and the errors associated with mediation by taking the concept to the next logical step: include document and service metadata that encodes the semantic intent. COIs which follow best practices for indexing and otherwise generating semantic metadata (see [R1047]) can reduce mediation issues. Semantic automation tools are emerging, such as the **Web Ontology Language** (**OWL**),[R1048] that assist in defining the semantic relationships and constraints in schemas.

These definitions can be used to automate the transformations between applications and services, to validate the transformations, and to support much more intelligent human-computer interaction. For example, a PEO C4I and Space sponsored program developed the Service Mediation Description specification for the DISA Net-Centric Capabilities Pilot.

This metadata document automatically generated user interfaces (input forms, data result tables, and map overlays) from semantically-described **Web services** and schemas, using a document format derived from WS-BPEL and other Web standards.

Best Practices

- BP1392: Register services in accordance with a documented service registration plan.
- BP1408: Use a **semantic** description language such as **Web Ontology Language** (**OWL**) or **Resource Definition Framework** (**RDF**) to represent an **Ontology**.
- BP1865: Provide sufficient program, project, or initiative **metadata** descriptions and automated support to enable **mediation** and translation of the data between **interfaces**.

P1063: Relational Database Management Systems

A **Relational Database Management System** (**RDBMS**) is a collection of data items organized as a set of formally-described tables. This permits accessing and reassembling data in many different ways without having to reorganize the database tables. It is important to ensure data quality and to access data quickly, using simple, easily understood dynamic queries. Towards these ends, an RDBMS offers such services as **triggers**, **stored procedures**, indices, constraints, **referential integrity**, efficient storage, and **high availability** features.

Database Independence

The **Structured Query Language** (**SQL**) allows for some portability of database access code when accessing various database products. It is important to use SQL standards that are open and well supported by database vendors and to avoid using proprietary extensions to the SQL standards. To further promote database independence, access the database only through **open standard** interfaces such as **Open Database Connectivity** (**ODBC**) or **Java Database Connection** (**JDBC**). This supports the goal of being able to swap out data sources and/or connect to multiple data sources without affecting the application or increasing software maintenance costs. Data-level adapters allow applications to access data through database calls that are native to the requesting application. At this point, the **business logic** can be shared with other data sources. This positions the application to move business logic from the database to the middle tier to support database independence.

Database Data Modeling

Data modeling is important for RDBMs as it improves database performance, improves the interoperability of the data, and allows for future growth and use of the RDBMS. The Data Modeling [P1003] perspective provides guidance for data modeling in addition to the guidance provided in this perspective.

Guidance

- G1014: Access databases through **open standard** interfaces.
- G1132: Implement the data tier using **commercial off-the-shelf** (**COTS**) **relational database management system** (**RDBMS**) products that implement a **Structured Query Language** (**SQL**).
- G1141: Base **data models** on existing data models developed by **Communities of Interest** (**COI**).
- G1144: Develop two-level database models: one level captures the **conceptual** or logical aspects, and the other level captures the **physical** aspects.
- G1146: Include information in the **data model** necessary to generate a **data dictionary**.
- G1147: Use **domain analysis** to define the constraints on input data validation.
- G1148: **Normalize** data models.
- G1151: Define declarative **foreign keys** for all relationships between tables to enforce **referential integrity**.
- G1151: Define declarative **foreign keys** for all relationships between tables to enforce **referential integrity**.
- G1153: Separate application, presentation, and data tiers.
- G1154: Use **stored procedures** for operations that are focused on the insertion and maintenance of data.
- G1155: Use **triggers** to enforce **referential** or data integrity, not to perform complex **business logic**.

Best Practices

- BP1139: Do not use proprietary **SQL** extensions.
- BP1140: Use SQL-2003 features in preference to **SQL-92** or **SQL-99**.
- BP1143: Use a **database modeling** tool that supports a two-level model (**Conceptual**/Logical and **Physical**) and **ISO-11179** data exchange standards.
- BP1145: Use vendor-neutral **conceptual/logical models**.
- BP1227: Do not allow installation of **MSMQ**-dependent clients.
- BP1248: Follow a naming convention.

- BP1249: Do not use generic names for database objects such as databases, schema, users, tables, views, or indices.
- BP1250: Use case-insensitive names for database objects such as databases, schema, users, tables, views, and indices.
- BP1251: Separate words with underscores.
- BP1252: Do not use names with more than 30 characters.
- BP1253: Do not use the **SQL:1999** or SQL:2003 reserved words as names for database objects such as databases, schema, users, tables, views, or indices.
- BP1254: For **command-and-control** systems, use the names defined in the Joint Command, Control and Consultation Information Exchange Data Model (JC3IEDM) for data exposed to the outside communities.
- BP1255: Use **surrogate keys**.
- BP1256: Use surrogate keys as the **primary key**.
- BP1257: Place a **unique key constraint** on the **natural key** fields.
- BP1258: Explicitly define the encoding style of all data transferred via **XML**.
- BP1259: Use indexes.
- BP1260: Define a **primary key** for all tables.
- BP1261: Monitor and tune indexes according to the response time during normal operations in the production environment.
- BP1262: In the case of Oracle, define indexes against the **foreign keys** (**FK**) columns to avoid contention and locking issues.
- BP1263: Gather storage requirements in the planning phase, and then allocate twice the estimated storage space.
- BP1264: For **high availability**, use hardware solutions when geographic proximity permits.

P1058: User Interfaces

The user interface represents all the components used to generate an interactive display that enables users to communicate with applications. The components of a user interface are not necessarily in the same physical location. For example user interface components are found both client side (as in the case of **HTML** pages) and server side (as in the case of components that generate HTML pages).

The following perspectives provide guidance for building user interfaces to promote interoperability of user interface components and improve human-computer interactions.

Detailed Perspectives

- Human-Computer Interaction [P1032]
- Browser-Based Clients [P1008]
- Thick Clients [P1074]

P1032: Human-Computer Interaction

Human-Computer Interaction (HCI) is the study, planning, and design of the interaction between humans and computers. HCI is a subset of Human Systems Integration (HSI). Human Systems Integration is a requirement for **Department of Defense** (**DoD**) acquisition; see Enclosure 8 of DoD Instruction 5000.02 [R1165]. In particular, this instruction requires that Program Managers shall take steps to include human factors engineering during system engineering over the lifecycle of the program to provide effective human-machine interfaces, "Where practical and cost effective, system designs shall minimize or eliminate system characteristics that require excessive cognitive, physical or sensory skills; entail extensive training or workload-intensive tasks; result in mission-critical errors; or produce safety or health hazards."

Interoperability includes both the technical exchange of information and the end-to-end operational effectiveness of that exchanged information as required for mission accomplishment. Whenever a user is required to interact with a computer user interface to accomplish a mission, and that interaction fails due to poor design (i.e., information is misunderstood or interaction results in a high cognitive load) then the risk of not accomplishing the mission is increased.

This perspective provides guidance and best practices that benefit human computer interaction to increase total system performance, reduce maintenance costs through better design, and accommodate the cognitive characteristics of the user. This perspective provides guidance for human factors common to all applications including data entry, data display, and user control appearance and behavior. The following detailed perspectives provide additional human factor guidance on more specific topics.

Detailed Perspectives

- Designing User Interfaces for Internationalization [P1112]
- Designing User Interfaces for Accessibility [P1111]
- Human Factor Considerations for Web-Based User Interfaces [P1108]

Guidance

- G1032: Validate all input fields.
- G1268: Label all data entry fields.
- G1270: Include scroll bars for text entry areas if the data buffer is greater than the viewable area.
- G1285: Use **relative font sizes**.
- G1286: Provide text labels for all buttons.
- G1287: Provide feedback when a transaction will require the user to wait.
- G1760: Solicit feedback from users on user interface usability problems.
- G1761: Provide units of measurements when displaying data.
- G1762: Indicate all simulated data as simulated.
- G1763: Indicate the security classification for all classified data.

Best Practices

- BP1054: Use conventional user interface controls that provide input choices for the user.
- BP1272: Disable dependent child controls when the parent control is inactive.
- BP1273: Gray out the push button label if a button is unavailable.
- BP1280: In tabular data displays, right justify integer data.
- BP1281: In tabular data displays, justify numeric data with decimals by using the decimal point.
- BP1290: Use a tool tip to display help information about a control when the purpose of the control is not self-evident.
- BP1291: Use obvious navigation controls for moving between pages in search results that span multiple pages.
- BP1298: Provide basic search functionality as the default with a link or button that provides more advanced search features.

- BP1767: Follow a standards-based process for human systems integration engineering.

P1112: Designing User Interfaces for Internationalization

Internationalization is the process of generalizing software so that it is interoperable with multiple languages (i.e., locales) and cultural conventions without the need for re-design or re-compilation. If an application designed for a U.S. audience will be used in combined or coalition warfare operations, it needs to provide a user interface that matches users' expectations, interacts with users in their native language, and displays data in a manner that is consistent with users' cultural conventions. The purpose of this perspective is to provide a starting reference for developers needing to support internationalization and provides best practices and resources.

Best Practices

- BP1764: Make all localizable user interface elements such as text and graphics externally configurable.
- BP1765: Declare the encoding type for all user interface content.
- BP1766: Develop user interfaces to accommodate variable syntactic structure for messages.

P1111: Designing User Interfaces for Accessibility

Section 508 of the Rehabilitation Act of 1973, as amended, requires that individuals with disabilities have access to and use of information that is comparable to that provided to federal employees and members of the public who are not disabled. The standards created under Section 508 define technology accessibility requirements for all types of information technology in the federal sector, including Web-based intranet and Internet information and applications.

Federal accessibility standards focus on providing redundancy in information presentation and interaction so individuals with disabilities can use different modalities to access information. The scope of Section 508 is confined to the federal sector, with a limited exemption for systems used for military command, weaponry, intelligence, and cryptologic activities. The exemption does not apply to routine business and administrative systems used for other defense-related purposes or by defense agencies or personnel. A Web application or portal that will be used in these systems is required to comply with Section 508 standards.

Guidance

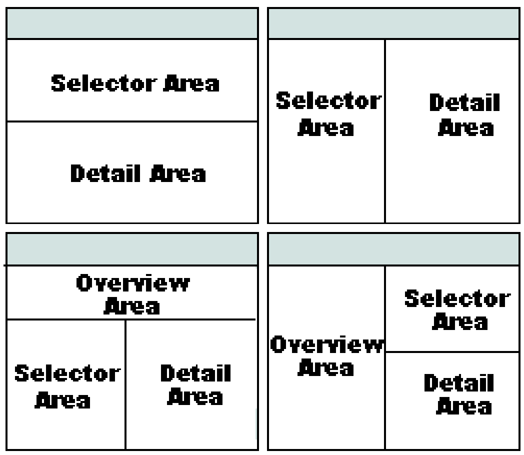
- G1044: Comply with Federal accessibility standards contained in Section 508 of the Rehabilitation Act of 1973 (as amended) when developing software user interfaces.

P1108: Human Factor Considerations for Web-Based User Interfaces

Web based user interfaces include **Web sites**, **Web applications**, and **Web portals**. This perspective provides guidance and best practices relating to human factors consideration that are specific to Web-based user interfaces. Additional information concerning general user interface guidance is available in the Human Computer Interaction [P1032] perspective.

Web sites tend to be content-centric and are generally developed using **HTML** for marking up content for Web pages. Sometimes other technologies such as **JavaScript** are used to add interactivity to Web pages. If developers choose to use a mix of HTML and other technologies to deliver Web content, it is important that they design their Web pages so the pages work correctly when viewed with browsers that support these technologies as well as with browsers that do not. In this way, all users will have an acceptable experience using the Web site.

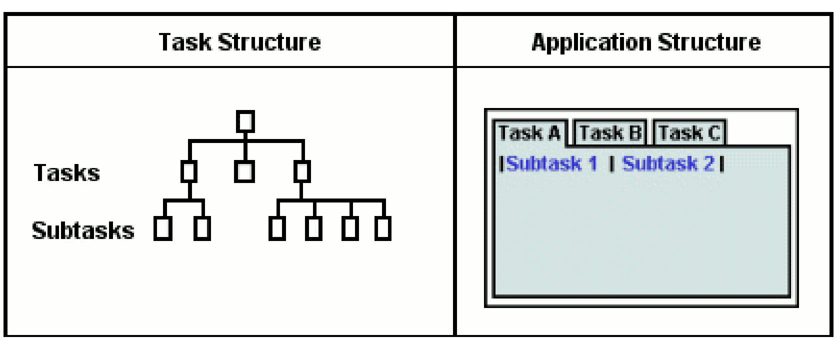
Web sites vary in their layout, but there are common themes for layouts that are widely used and understood users. Some example Web site layouts are shown in this figure:



I1178

Web Applications

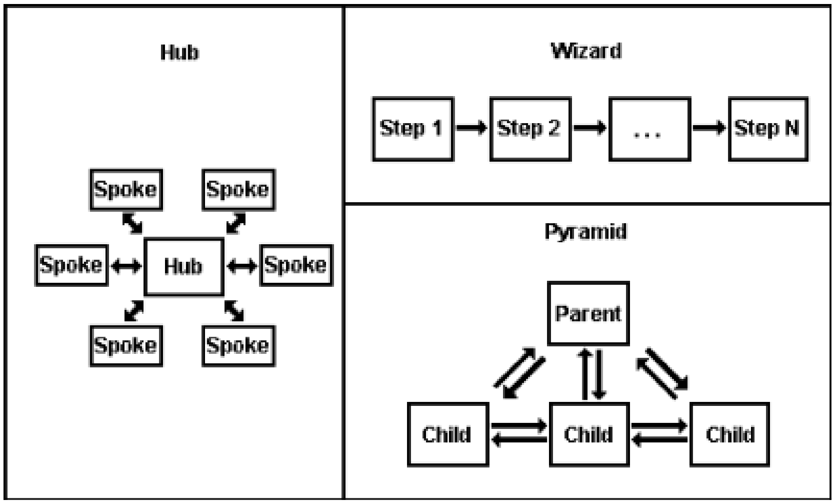
A Web site tends to be content-centric, but a Web application tends to be task-centric and organizes content around a hierarchy of tasks. An example user interface for a given task structure is shown in this figure:



I1179

A Web application often supports interactivity similar to that available in a desktop application but delivered to users within the framework of a browser. Because a Web application allows users to create, save, and delete data, it supports greater complexity in design and interactivity compared to a content-oriented site.

In addition to application structure, there are common navigation models that are well understood by users for Web application workflow. Some common examples are in this figure:



I1180

The "hub navigation metaphor" is often used for applications where a task consists of multiple independent steps that are performed in any order. The hub page present users with a collection of "spoke" pages that they access from a single page; when users submit their input, they are returned to the hub page.

The "wizard navigation" metaphor is often used when a task consists of multiple interdependent steps that are performed in a predefined order. In this metaphor, a wizard presents users with a collection of pages that they interact with sequentially; when the user submits their input, the user is presented with the next page.

The "pyramid navigation" metaphor is often used when it is important to navigate to sibling, child, or parent pages while completing tasks; when the user submit their input, they are returned to the same page where they follow links to another adjacent page in the pyramid.

Web Portals

A portal is a type of Web application that provides a gateway from which users can access the information, resources, and services they need. A portal aggregates and organizes content from different sources within a Web page related to specific mission or business task. Sometimes a portal allows users to personalize what and how information is presented to them such as selecting and arranging the content presented on the portal page and to choosing the "look and feel" of the display.

The pages in a portal contain portlets that enable users to view and/or interact with Web-based information related to a specific function. A portlet provides more than a view of existing Web content, functioning instead as a complete application with multiple states and view modes.

Since portals are designed to contain portlets from various sources, it is important for portlet developers to develop portlets carefully to allow for a standard presentation and behavior when the portlet is deployed within the portal. Allowing for configuration for presentation such as fonts and colors allows for a common look and feel across all portlets within a portal. Developing portlets according to standards for user controls enables a better experience for the end user with respect to common portlet control behavior.

Guidance

- G1267: Use **HTML** data entry fields on **Web pages**.
- G1276: Do not modify the contents of the Web browser's status bar.
- G1277: Do not use tickers on a Web site.
- G1278: Use the browser default setting for links.
- G1284: Use only one font for **HTML** body text.
- G1292: Use text-based Web site navigation.
- G1294: Provide a site map on all Web sites.
- G1295: Provide redundant text links for images within an **HTML** page.

- G1566: Use **`alt`** attributes to provide alternate text for non-text items such as images.
- G1759: Use a style guide when developing Web portlets.

Best Practices

- BP1038: Use a **sans serif** font (e.g., Arial, Verdana) in Web pages rather than a serif font (e.g., Times New Roman).
- BP1039: Do not underline any text unless it is a link.
- BP1041: Do not change the default colors of the links.
- BP1042: Do not build a **Web page** where the horizontal width is greater than the screen (vertical scrolling is fine), planning for the lowest common denominator to be super-VGA resolution (800 x 600).
- BP1297: Structure a Web site hierarchy so users can reach important information and/or frequently accessed functions in a maximum of three jumps.
- BP1299: Include a link back to the home page on all Web pages.
- BP1768: Use design patterns for application navigation.

Part 5: Developer Guidance > User Interfaces > Browser-Based Clients

P1008: Browser-Based Clients

This perspective provides guidance for creating and interfacing to thin clients. It includes links to the perspectives included in the list in the following subsection.

Detailed Perspectives

- XML Rendering [P1084]
- Active Server Pages (Classic ASP) [P1001]
- Active Server Pages for .NET (ASP.NET) [P1002]
- JavaServer Pages (JSP) [P1040]
- Web Portals [P1077]
- Style Sheets [P1070]

Guidance

- G1043: Separate formatting from data through the use of **style sheets** instead of hard coded **HTML** attributes.
- G1271: Provide instructions and **HTML** examples for all style sheets.
- G1283: Use **linked style sheets** rather than embedded styles.

Best Practices

- BP1040: Use hex codes for all colors (e.g., #FFFF33), never the color name (e.g., yellow).
- BP1291: Use obvious navigation controls for moving between pages in search results that span multiple pages.
- BP1567: Use the <**`abbr`**> and <**`acronym`**> tags to specify the expansion of acronyms and abbreviations.
- BP1568: Use a markup language to represent mathematical equations within Web pages.

Part 5: Developer Guidance > User Interfaces > Browser-Based Clients > XML Rendering

P1084: XML Rendering

XML can render display-device-neutral output to a particular output device given a set of display rules or a **style sheet**. The **XSLT** file is the decoupled output formatter that determines how the output device renders the data.

Guidance

- G1045: Separate **XML** data presentation **metadata** from data values.

P1001: Active Server Pages (Classic ASP)

Microsoft Active Server Pages (ASP) is a server-side scripting environment for creating dynamically-generated **Web** content (generally but not limited to **HTML** pages).

ASP supports combining static content (such as HTML, **CSS**, and images) with server-side-executed scripts and Component Object Model (COM) objects to generate interactive content. ASP supports multiple scripting languages including **Visual Basic Scripting** (**VBScript**) and **JScript**.

ASP is generally superseded by ASP.NET; therefore; ASP is sometimes referred to as ***Classic ASP***.

Guidance

- G1050: In Active Server Pages (Classic ASP), isolate the presentation tier from the middle tier using Component Object Model (COM) objects.
- G1058: Use the Model, View, Controller (MVC) pattern to decouple presentation code from other tiers.

P1002: Active Server Pages for .NET (ASP.NET)

Microsoft Active Server pages for .NET (ASP.NET) is a server-side environment for creating dynamically-generated **Web** content (to include but not limited to **HTML**, **XML**, and **Web services**). ASP.NET supports multiple programing languages through its foundation on the **.NET Framework** and its **Common Language Runtime** (**CLR**).

ASP.NET supersedes the older Active Server Pages, now often called ***Classic ASP***. ASP.NET offers the following advantages over Classic ASP:

- ASP.NET supports the separation of presentation layer with the business logic through its support of a code-behind feature
- Precompiled code allows for ASP.NET to have performance benefits over Classic ASP
- Precompiled code allows for ASP.NET developers to trap coding errors sooner in the lifecycle (as opposed to runtime with Classic ASP)
- ASP.NET supports improved exception handling
- ASP.NET supports object-oriented design programming

Guidance

- G1052: Use the code-behind feature in ASP.NET to separate presentation code from the business logic.
- G1053: Do not embed HTML code in any code-behind code used by aspx pages.
- G1056: Specify a versioning policy for **.NET** assemblies.
- G1058: Use the Model, View, Controller (MVC) pattern to decouple presentation code from other tiers.

Part 5: Developer Guidance > User Interfaces > Browser-Based Clients > JavaServer Pages (JSP)

P1040: JavaServer Pages (JSP)

JavaServer Pages (**JSP**) technology is a **Java** technology that supports software and **Web** developers in creating dynamic server-generated content (including but not limited to Web content). JSP syntax supports mixing delimited blocks of Java code within static content (such as **HTML** and **CSS**) in a **Web page**. The resulting page is compiled and executed server-side allowing for dynamic Web content.

This mixture of Java code within static content often leads to increased development and maintenance costs as one person is often responsible for the layout, static content, and design elements in a JSP page while another person is often responsible for the Java code within the same JSP page. The ***JavaServer Pages Standard Tag Library (JSTL)*** helps address separation of concerns by allowing for the separation of business logic into tag libraries, which are referenced within the static Web content. Using JSTL helps support the use of the ***model-view-controller*** pattern which allows for the responsible parties to maintain better their respective portions of a JSP page.

Guidance

- G1058: Use the Model, View, Controller (MVC) pattern to decouple presentation code from other tiers.
- G1060: Encapsulate Java code in JavaServer Pages Standard Tag Libraries (JSTL) when using the code in **JavaServer Pages** (**JSP**).

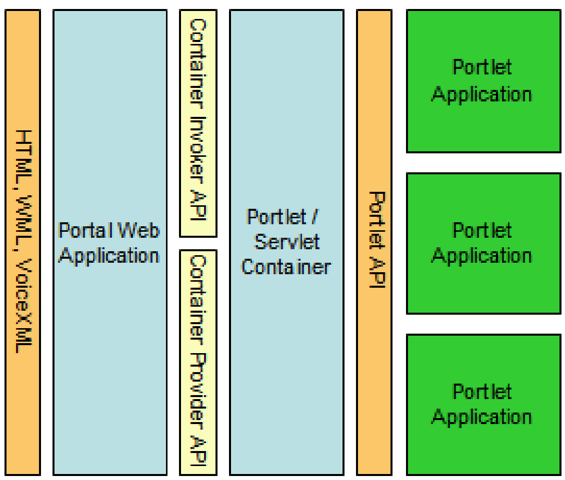
P1077: Web Portals

A **Web portal** is a **Web site** that provides a starting point or gateway to other resources on the Internet or an intranet. Access to a Web portal is typically via **HTTP** and can be in any number of formats including **HTML**, **Wireless Markup Language** (**WML**) or **VoiceXML**. A Web Portal often uses a **Web Application** that provides **single sign-on**, content **integration** and **aggregation** from different sources, **collaboration**, content and document management and personalization of the presentation. It hosts the presentation layer of different backend systems in a **single touch point**.

An attractive feature of a **portal** to an **enterprise** is to aggregate different applications into a single **page** with a common **Look and Feel** that enhances the portal **end user's** experience. A portal may also have sophisticated personalization features, which provide customized content to individual end users or to their roles within the enterprise. **Portal pages** can dynamically coordinate different **portlets** to create specialized content for different portal end users.

IBM's Websphere depicts the basic architecture of portals as a series of layers between the end user's environment such as **browsers**, mobile devices and phones. The portal processes an end user **client** request. A Web Application that interacts with the portlet to request the web page for the current end user is produced. The portal Web Application then uses the **portlet container** for each portlet to retrieve the requested content through the **Web Container Invoker API**. The portlet container calls the portlets through the Portlet API. The Container Provider Service Provider Interface (SPI) enables the Web Application to retrieve information from the portal through its portlet container.

The portlet container invokes the portlets, provides a runtime environment, and manages the lifecycle of the portlet. In addition, it provides persistence for the portlet to store end user information enabling the production of customized Web pages.



I1006

Guidance

- G1245: Isolate the **Web service portlet** from platform dependencies using the **Web Services for Remote Portlets** (**WSRP**) Specification protocol.

Best Practices

- BP1246: Base Java-based portlets on **JSR 168**.
- BP1247: Encapsulate Java-based **portlets** in a Web Application Archive (.war) file.

P1070: Style Sheets

A **style sheet** is a template used to customize the layout of a **Web site**. Style sheets allow Web sites to present content in a consistent manner. Web designers can create custom tags to override default values:

```
h1,h2,h3 {
   font-family: verdana, arial, 'sans serif';
}
p,table,li {
   font-family: verdana, arial, 'sans serif';
   margin-left: 10pt;
}
```

Guidance

- G1043: Separate formatting from data through the use of **style sheets** instead of hard coded **HTML** attributes.
- G1271: Provide instructions and **HTML** examples for all style sheets.
- G1283: Use **linked style sheets** rather than embedded styles.

Best Practices

- BP1038: Use a **sans serif** font (e.g., Arial, Verdana) in Web pages rather than a serif font (e.g., Times New Roman).
- BP1040: Use hex codes for all colors (e.g., #FFFF33), never the color name (e.g., yellow).
- BP1041: Do not change the default colors of the links.

P1074: Thick Clients

A thick client (often called "fat client") is a client machine in a client/server environment that performs most or all of the application processing with little or none performed in the server. Developers should use existing user interface (UI) toolkits rather than build their own; the Sun Developer Network *Java SE Desktop Overview*[R1078] provides information on two such toolkits for Java (Swing and AWT).

Guidance

- G1030: Use a user interface **component** library.

P1052: Middleware

Middleware is software that resides between software components and the underlying platforms (i.e., operating system, services, databases, hardware, etc.) with which the software interacts. Middleware enables communication between different, often distributed, software components while isolating integration complexity into well-understood software abstractions. Middleware is sometimes referred to as "plumbing" as it connects two or more components allowing for the flow of data between them.

> ***Note:*** *The distinction between operating system and middleware functionality is sometimes arbitrary. Some functionality previously provided by middleware is now integrated within operating systems. For example, modern operating systems commonly include the TCP/IP stack as well as support for playback of audiovisual content.*

Middleware often supports and simplifies the development of complex, distributed software systems in a **Service-Oriented Architecture** (**SOA**). Middleware often is in many places within a software architecture (for example client-side as well as sever-side software components) and includes software abstractions from a variety of categories:

- Messaging middleware (sometime called **Message-Oriented Middleware**) supports publishing, consuming, routing, and processing of messages
- Service-related middleware supports the production, hosting, management, and interaction with services implemented using a variety of technologies
- **Enterprise Service Bus** (**ESB**) middleware provides invocation, messaging, mediation, transport, management, and security in a SOA environment
- Embedded middleware provides services, communication, and hardware abstraction on embedded platforms such as software radios
- **Remote Procedure Call** (**RPC**) middleware provides access to procedures running on remote systems
- **Object Request Broker** (**ORB**) middleware enables software applications to send objects and request services in an object-oriented fashon
- Data access middleware such as **Enterprise JavaBeans** (**EJBs**) provides abstraction between software and underlying data storage solutions
- **Application server** middleware that facilitate developing and deploying distributed enterprise applications

Programming environments such as **.NET** and **Java EE** often act as middleware, providing abstractions and **Application Programming Interfaces** (**APIs**) between software and underlying platforms.

Detailed Perspectives

- Messaging [P1047]
- Web Services [P1078]
- Enterprise Service Bus (ESB) [P1389]
- Software Communication Architecture [P1087]
- CORBA [P1011]
- .NET Framework [P1086]
- Java EE Deployment Descriptors [P1037]

P1047: Messaging

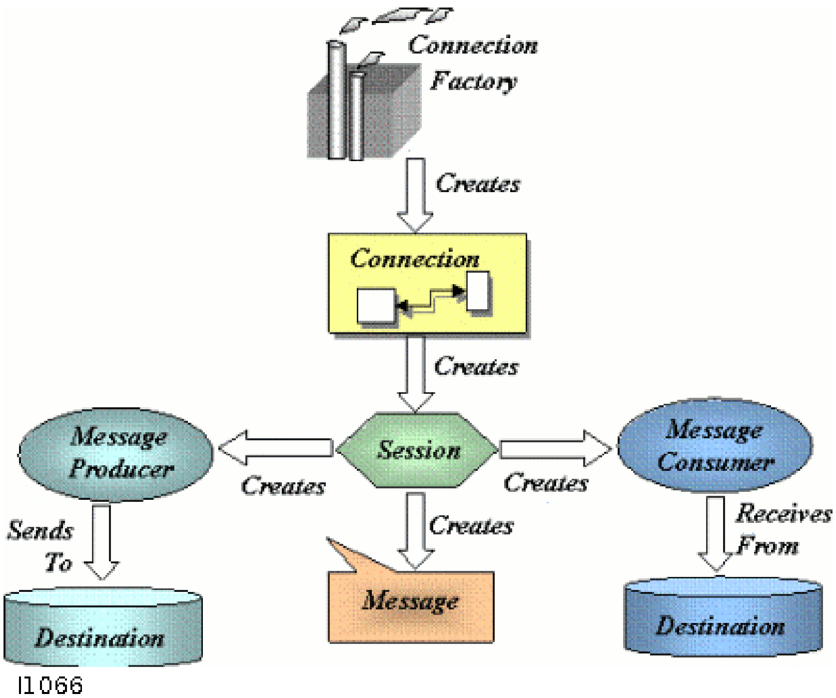
The explosion of the Internet required applications to communicate and interoperate with other applications and services. Messaging systems play an important role in enterprise applications because computers and networks are inherently unreliable and messaging systems are perfectly suited to operate in disconnected environments. They provide a reliable, secure, event-driven message-delivery communication mechanism. Unlike traditional **RPC**-based systems (**RMI** or **CORBA**), most message-oriented based systems operate peer-to-peer.

The messaging paradigm offers three major advantages:

- Allows applications to communicate asynchronously. This means the system sending the **message** does not have to wait around for a response.
- Provides more robustness and reliability; messages do not get lost if a **client** has crashed or is unavailable.
- Multiplexes messages and sends them to multiple clients.

There are other advantages such as transactional message support, message prioritization, load balancing, and firewall **tunneling**. However, these features usually depend on how the **Message-Oriented Middleware** (**MOM**) is implemented.

This diagram shows the relationship of the classes and interfaces in the **Java Message Service** (**JMS**) **API**. Developers use these classes and interfaces to create a JMS application.



Detailed Perspectives

- Message-Oriented Middleware (MOM) [P1046]
- Data Distribution Service (DDS) [P1190]
- Messaging with MSMQ [P1048]

P1046: Message-Oriented Middleware (MOM)

Message-oriented middleware acts as an arbitrator between incoming and outgoing **messages** to insulate producers and consumers from other producers and consumers A **MOM** typically is implemented using proprietary **protocols** and interfaces, which means that different implementations are usually incompatible. Using a single implementation of a MOM in a system typically leads to dependence on the MOM vendor for maintenance, support, and future enhancements. Maturing standards such as **Java Message Service** (**JMS**) and **SOAP Web services** are reducing vendor dependencies by standardizing message content and providing standard interfaces to the various MOM **APIs**.

Advantages

- A MOM provides a common reliable way for programs to create, send, receive, and read messages in any distributed enterprise system.
- A MOM ensures fast, reliable, asynchronous communications, guaranteed message delivery, receipt notification, and transaction control.
- A MOM increases the interoperability, portability, and flexibility of an application by allowing it to be distributed over multiple heterogeneous platforms.
- A MOM enables applications to exchange messages with remote programs without having to know on what platform or processor the other application resides.

Disadvantages

- A MOM does not help with interoperability directly, as applications need to agree on message content and format at development time.
- The current marketplace is filled with proprietary implementations of features, so moving between MOMs usually requires recoding; JMS and other standard interfaces help in this area but do not usually cover all of the vendor's extended functionality.

Features

Guaranteed message delivery	MOMs provide a message queue between interoperating processes. If the destination process is busy or offline, the message is held in a temporary storage location until it can be processed.
Asynchronous and synchronous communications	MOMs allow multitasking. Once an application sends out a message to a receiving application, the MOM allows the **client** application to handle other tasks without waiting for a response from the receiving application. Supports blocking method calls.
Transaction support	Most MOMs support transactions.
One-time, in-order delivery	MOMs guarantee that each message will be delivered once and that messages are received in the order in which they are sent.
Message routing services	MOMs support least-cost routing and can reroute around network problems.
Notification Services	MOMs provide audit trails, journaling, and notifications when messages are received.

Message Models

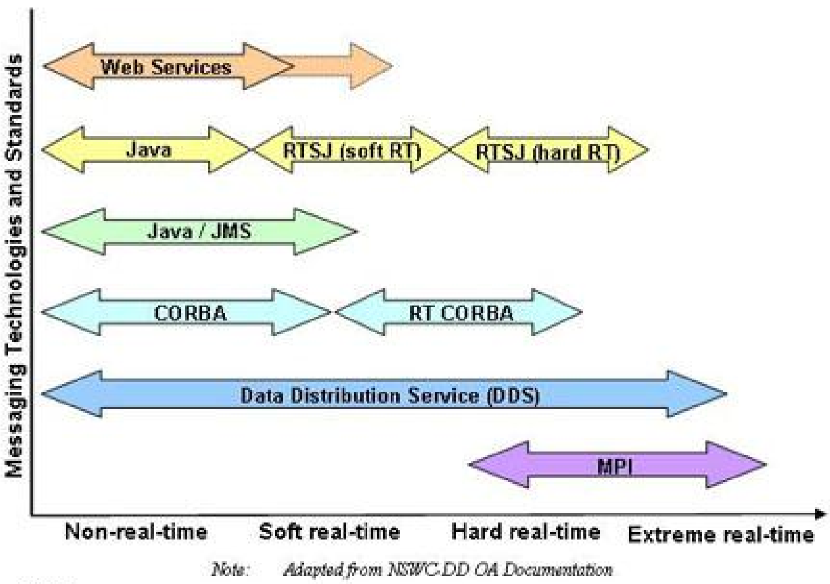
The most important aspect of a message-based communication system is the message. The most common messaging models are the following:

- Point-to-Point (p2p)
- Publish/Subscribe (pub/sub)
- Request-Reply

P1190: Data Distribution Service (DDS)

Data Distribution Service for Real-time Systems (**DDS**) is an **Object Management Group** (**OMG**) specification for distributing data messages using the **Publish-Subscribe** design pattern. It defines a common **application programming interface** (**API**) that cleanly separates the data distribution functionality from the application functionality. DDS also simplifies the complexity associated with application programming by separating the details of publishing data messages from those for subscribing to data messages using a **Quality of Service** (**QoS**) approach. The implementation of the interface effectively creates a data distribution service that applications can access.

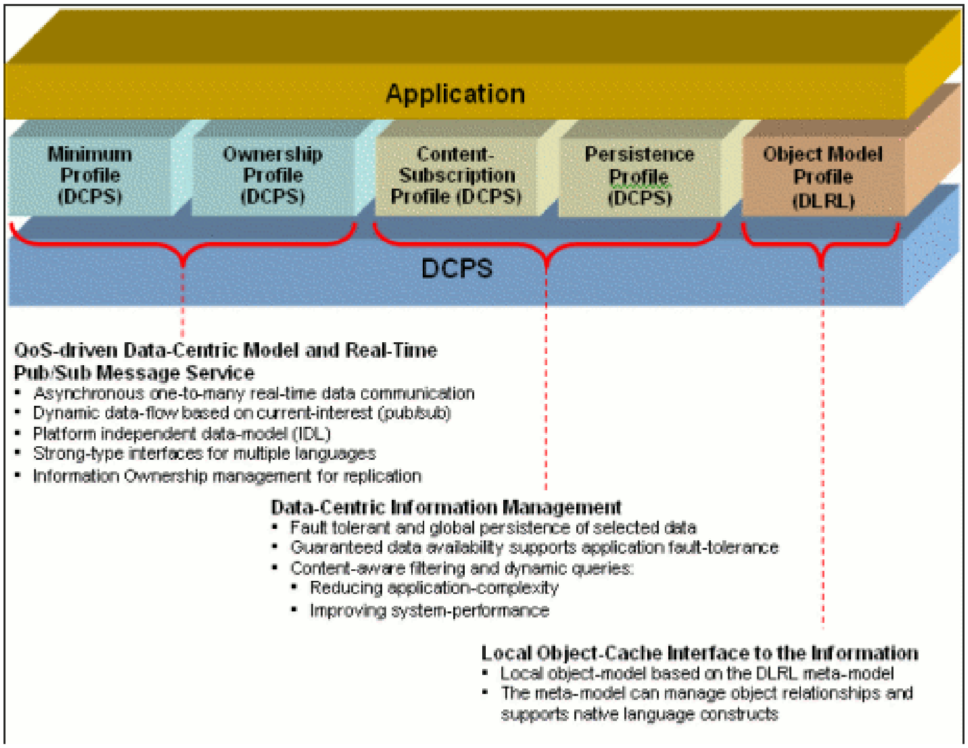
The use of QoS makes DDS especially appealing as an integration middleware in heterogeneous systems. DDS QoS allows fine-grained tuning of the properties for each information flow including the lowest level data writer and data reader. Therefore, the system can devote its resources to the more critical flows ensuring they are achievable. Also, the use of QoS combined with the inherent real-time nature of the DDS allows DDS solutions to span the complete spectrum from Enterprise (non-real-time) to hard real-time applications as shown in the following figure.



I1195

DDS Profiles

The specification divides the complexity of the full data distribution functionality into five profiles (Minimum, Ownership, Content Subscription, Persistence, and Object Model) to help applications meet their individual requirements. The applications can use any or all of the profiles to access the Data Distribution Service.



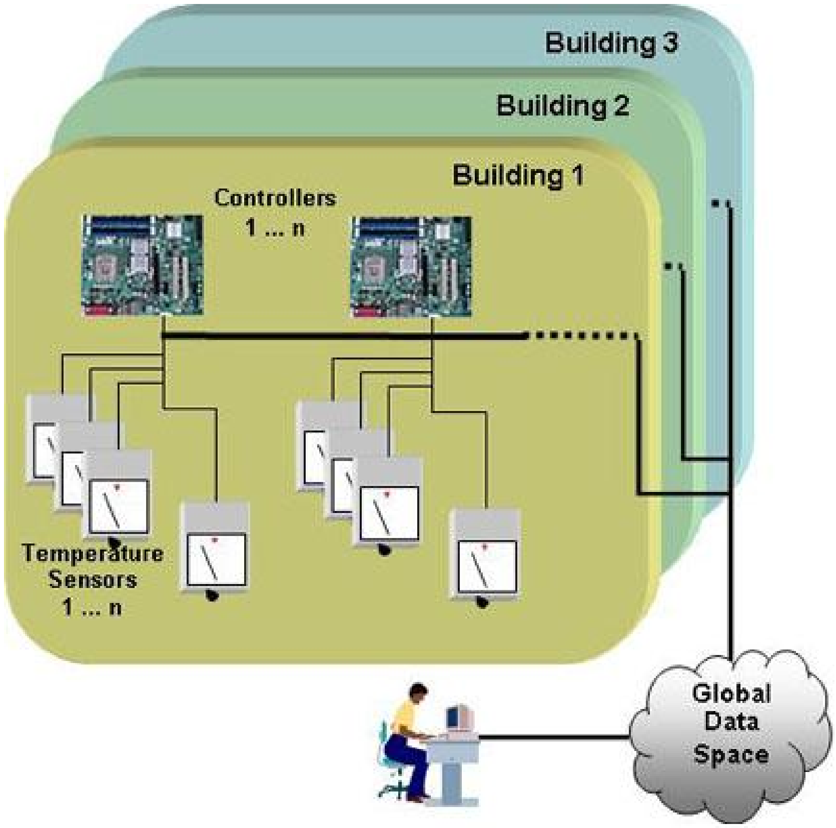
I1196

DDS Compliance Profiles

Minimum	This profile contains just the mandatory features of the **DCPS** layer. None of the optional features are included.
Ownership	This profile adds the following: • the optional setting **`EXCLUSIVE`** of the **`OWNERSHIP`** kind • support for the optional **`OWNERSHIP_STRENGTH`** policy • the ability to set a depth > 1 for the **`HISTORY`** QoS policy.
Content-Subscription	This profile adds the optional classes **`ContentFilteredTopic`**, **`QueryCondition`**, and **`MultiTopic`**. This profile also enables subscriptions by content.
Persistence	This profile adds the optional QoS Policy **`DURABILITY_SERVICE`** as well as the optional settings **`TRANSIENT`** and **`PERSISTENT`** of the **`DURABILITY`** QoS Policy kind. This profile enables saving data into either transient memory, or permanent storage so that it can survive the lifecycle of the **`DataWriter`** and system outings.
Object Model	This profile includes the **DLRL** and also includes support for the **`PRESENTATIONaccess_scope`** setting of **`GROUP`**.

Example

The following diagram depicts using a data-oriented approach to solve a typical distributed system problem. The goal in this example is to maintain the temperature in many buildings, using embedded controllers each connected to a number of sensors. Each of these sensors and control processes are connected through a transport mechanism such as Ethernet and use basic protocols such as **TCP**-**UDP**/**IP** to provide standardized communication.



I1197

To achieve data integrity and fail-over capabilities, multiple controllers and sensors are deployed in each building. Controllers within a building collaborate in the process of collecting data from the various sensors. Applications access and manipulate the data through the use of a global data space.

Data-centric technologies such as databases and Service-Oriented Architecture **Web service**-based applications can interoperate seamlessly with the embedded sensors. These technologies provide a standards-based way for external applications to get, process and manipulate real-time sensor data with out having to know the specifics of the real-time data infrastructure. Furthermore, decoupling the data from the technology that manipulates the data contributes to developing a truly data-centric application. In this example, the external access and monitoring applications can simply receive real-time updates from any sensor as well as issue commands to the various controllers via DDS, **SQL**, etc., to maintain suitable temperatures.

Data Model

For simplicity, this example will focus on the data the sensors send to their controller and how they can be distributed throughout the entire system. The first step in a data-centric approach is to describe the data format carefully in a standards-based way, either IDL or XML, and give it a **Topic** name. Topics are the element of the DDS middleware publish-subscribe standard which identify the data objects and provide the basic connection between **publishers** and **subscribers**. Subscribers (the Controllers in this example) register Topics with the middleware that they wish to receive. Publishers (the individual sensors in this example) register Topics with the middleware that they will send. If the Topics do not match, effective communication does not take place.

Topics enable one to find specific information sources when architecting a loosely coupled system; that is, one which does not know a priori how many sensors or controllers there are going to be or where they all are. The Controller can simply subscribe to **`TempSensor`**, the Topic's name, and receive all the sensor updates for that building. Similarly, a sensor does not need to know if it is sending its data to one or multiple Controllers or even an external data store.

Specification of the Topic's name is a key element in a **data-centric** approach to creating open **real-time systems**. One could name each sensor's Topic based on its unique location in the building, **`Floor12Room3Sensor14`** for example, but the Controller would then need to be configured every time a sensor is added or removed from the system. Topics (name and type) define the standard interface for the distributed system; chose them appropriately.

Data Type

Specification of the Topic's data type is equally important as the Topic's name. DDS specifies the use of a subset of the **Interface Definition Language** (**IDL**) for specifying a Topic's data type.

> ***Note:*** *IDL readily maps to XML and SQL semantics.*

```
struct SensorData
{
  long    id; //@key
  float   temp;
};
```

I1198

In the definition of the Topic's type, chose one or more data elements to be a ***Key***. Keys provide scalability and the communication infrastructure can use the key to sort and order data from many sensors. In this example, without keys, one would need to create individual Topics for each sensor. Topic names for these topics might be **`Sensor_1`**, **`Sensor_2`**, and so on. Therefore, even though each Topic is comprised of the same data type, there would still be multiple Topics. With keys, there is only one topic, **`TempSensor`**, used to report temperatures.

New sensors can be added without creating a new Topic. The publishing application would just need to set a new id when it was ready to publish. An application can also have a situation where there are multiple publishers of the same Topic with the same key defined. This enables the application to provide redundancy. Per this example, two sensors in the same room using the same Key value will measure the same piece of information. Managing the redundancy, should one or both sensors report to the controller, is accomplished though Quality-of-Service (QoS).

Domains and Partitions

A **Domain** is the basic DDS construct used to bind individual **publications** and **subscriptions** together for communication. A distributed application can elect to use single or multiple DDS Domains for its data-centric communications. A Partition is a way to separate Topics logically within a DDS Domain.

In the context of the example, Partitions can group sensors on different floors. For example, to divide the building into different zones where each zone is controlled by a dedicated Controller, the Sensor and Controller could set the Partition to **`Floor 1`** and **`Floor 1-6`**, respectively. The Controller will receive data from all Sensors on Floors 1 through 6. Using Partitions makes it easy to group which Sensors are ***hooked*** to a Controller and a Controller can take over a different zone by changing or adding to its Partition list.

In the example, different buildings map to different DDS Domains. Domains isolate communication, promote scalability and segregate different classifications of data.

Quality of Service

The following briefly details how one might leverage a few of the DDS QoS Policies for this example.

Ownership

The Ownership QoS specifies whether or not multiple publishers can update the same data object and is how to achieve fault-tolerance using DDS.

Returning to the example, having multiple sensors in the same room and only wanting to get data from the primary (as long as it is functioning), then the Ownership QoS policy is set to Exclusive, stating that only one sensor can update that keyed value. Setting the Ownership QoS value to Shared indicates that there can be multiple sensors in the same room all reporting the same piece of keyed data. In this case the Controller would get all updates from all sensors and treat the values as the same measurement.

Durability

The Durability QoS specifies whether past samples of data will be available to newly joining subscribers.

Considering the example, if a Controller were to reboot, rather than require all sensors to resend their data, or require the data to be sent at a periodic rate in case the systems reboots, one simply gets the latest published value for every attached sensor. This effectively decouples the system in time and provides a high degree of data integrity.

History

History specifies how many data samples are stored for later delivery.

In the case of the example, a rebooted controller may want the last 5 samples from its sensors, so that it can make sure that readings are consistent.

Reliability

The Reliability QoS may be set on a per Topic basis and informs the middleware that the Subscription should receive all data (no missed samples) from a Publication even over non-reliable transports. Generally for periodic publications Reliability doesn't need to be set, since it can just get the updated value one sample period later. Although periodic sensor data doesn't need to be delivered reliably, synchronization commands between Controllers in this example could be.

Summary

This simply stated example is surprisingly complex, containing many elements of real-time messaging, data integrity and failover capabilities, integration with databases, web services, as well as scalability and modularity concerns while remaining data-centric.

Detailed Perspectives

- Decoupling Using DDS and Publish-Subscribe [P1191]
- DDS Quality of Service [P1192]
- DDS Data-Centric Publish-Subscribe (DCPS) [P1193]
- DDS Data Local Reconstruction Layer (DLRL) [P1197]

P1191: Decoupling Using DDS and Publish-Subscribe

A fundamental tenet of data-centricity and **DDS** is the decoupling between information providers and consumers. The decoupling is conceptually anonymous in that the producers do not need to know who the consumers are, and similarly the consumers do not need to know who the producers are. They are in fact each communicating independently using the DDS **Domain** (i.e., **Global Data Space**). Persistence services in the Global Data Space allow data written by an application to be available to late joining applications, even if the original application is no longer present.

While communications can precede anonymously, DDS does offer the means for an application to detect its communication partner. A ***Writer*** can see who the matched Readers are, and similarly a ***Reader*** can identify the matched Writers. If so requested, the application is given notification of new matches and can even "veto" specific Readers or Writers.

Decoupling and anonymity is accomplished using the publish-subscribe paradigm. Applications that want to provide information indicate their intent to publish by creating a **DataWriter** and specifying the offered **Quality of Service** (**QoS**) and a **Listener**. Applications that want to access information indicate their intent to subscribe by creating a **DataReader** and specifying the requested QoS and a Listener.

Publishers are matched with **subscribers** by DDS using the **Topic** and the QoS, and DDS automatically sets up the needed communication paths and resources such that information (data updates) can flow directly with the highest possible performance. **Listeners** are used to indicate to the application that certain events of interest have taken place, such as the arrival of new information for **`DataReaders`**, violations in the QoS contracts, matching of new Publishers/ Subscribers or other middleware-observed events.

QoS contracts provide the means for applications/components to remain modular and independent from each other while at the same time having some control over how the information is provided or delivered. For example, a reading application may have some minimum requirements regarding reliability, ordering, coherence, or frequencies of updates, and a writing application may have some resource limits with regards to how much history it can maintain or how many readers it can handle. The QoS contract can specify these requirements and DDS checks and monitors them. In addition QoS can configure resources, message priorities, history, etc. The ability to fine-tune separately the behavior of each **`DataWriter`** and **`DataReader`** is one of the reasons why DDS can span the range from real-time to near-time to enterprise systems.

Guidance

- G1802: Catch **Data Distribution Service** (**DDS**) events.
- G1807: Check the return values of **Data Distribution Service** (**DDS**) functions.
- G1809: Handle all **Data Distribution Service** (**DDS**) events using one of the **subscriber access APIs**.
- G1810: Use **data models** to document the data contained within the **Data Distribution Service** (**DDS**) **Data-Centric Publish Subscribe** (**DCPS**).

Best Practices

- BP1811: Isolate all use of vendor specific extensions to the **Data Distribution Service** (**DDS**).
- BP1825: Use the **`ignore_participant`** operation on the **DomainParticipant** to deny access to another DomainParticipant trying to join a **Data Distribution Service** (DDS) **Domain**.
- BP1827: Use the **`ignore_publication`** and **`ignore_subscription`** on the **DomainParticipant** to deny access to a **Data Distribution Service** (DDS) **Topic** by a specific **DataWriter** or **DataReader**.
- BP1830: Use the **Data Distribution Service** (DDS) Content Profile to tailor subscription message data.
- BP1831: Use the **Data Distribution Service** (DDS) Persistence Profile to ensure durable data delivery.

P1192: DDS Quality of Service

Quality of Service (**QoS**) is a general concept that specifies the behavior of a service. Programming service behavior by means of QoS settings offers the advantage that the application developer only indicates what is wanted rather than how to achieve the specific QoS. Generally speaking, QoS is comprised of several QoS policies. Each QoS policy is then an independent description that associates a name with a value. Describing QoS by means of a list of independent QoS policies gives rise to more flexibility.

> ***Note:*** *As **Service-Oriented Architecture** (**SOA**) systems evolve and become richer in the number of publishers and subscribers supported with time, the use of well defined and specific QoS parameters becomes essential in managing the complexity of the system and the loosely coupled nature of the services.*

Data-centric communication using **DDS** provides the ability to specify various parameters like the rate of publication, rate of subscription, how long the data is valid, and many others. These QoS parameters allow system designers to construct a distributed application based on the requirements for, and availability of, each specific piece of data. A data-centric environment allows a communication mechanism that is custom tailored to the distributed application's specific requirements yet remains a loosely coupled design and architecture.

The ability to set QoS on a per-entity basis is a significant capability provided by DDS. Being able to specify different QoS parameters for each **Topic**, **Publisher** or **Subscriber** gives developers many options when designing their systems. Through the combination of these parameters, a system architect can construct a distributed application to address an entire range of requirements, from simple communication patterns to complex data interactions.

Guidance

- G1771: Explicitly define the **Data Distribution Service** (**DDS**) **Quality of Service** (**QoS**) Policies to describe the behavior of a **publisher**.
- G1801: Explicitly define a **Topic Quality of Service** (**QoS**) for each **Data Distribution Service** (**DDS**) Topic within a DDS **Domain**.
- G1803: Explicitly define the **Data Distribution Service** (**DDS**) **Quality of Service** (**QoS**) Policies to describe real-time messaging criteria for **Publishers**.
- G1804: Explicitly define the **Data Distribution Service** (**DDS**) **Quality of Service** (**QoS**) Policies to describe **DataWriter**.
- G1805: Explicitly define the **Data Distribution Service** (**DDS**) **Quality of Service** (**QoS**) Policies to describe the behavior of the **Subscriber**.
- G1806: Explicitly define the Request-Offered **Data Distribution Service** (**DDS**) **Quality of Service** (**QoS**) Policies to describe the behavior of the **DataReader**.
- G1808: Handle all **Data Distribution Service** (**DDS**) **Quality of Service** (**QoS**) contract violations using one of the **Subscriber access APIs**.

Best Practices

- BP1812: Use the **`RELIABILITY`** **Quality of Service** (**QoS**) kind **`BEST_EFFORT`** for **Data Distribution Service** (**DDS**) **Topics** that are written frequently where missing an update is not important because new updates occur soon thereafter.
- BP1813: Use the **`RELIABILITY`** **Quality of Service** (QoS) kind **`RELIABLE`** for **Data Distribution Service** (DDS) **Topics** written sporadically or where it is important that the current data in the Topic is received reliably.
- BP1814: Use the **`DEADLINE`** **Quality of Service** (QoS) to for **Data Distribution Service** (DDS) **DataWriters** for which data is published at a constant rate.
- BP1815: Use the **`DEADLINE`** **Quality of Service** (QoS) for **Data Distribution Service** (DDS) **DataReaders** that expect data to be sent to them at a constant rate.
- BP1816: Use the **`LIVELINESS`** **Quality of Service** (QoS) for **Data Distribution Service** (DDS) **Topics** where data is not sent sporadically; that is, it is sent with no fixed period.

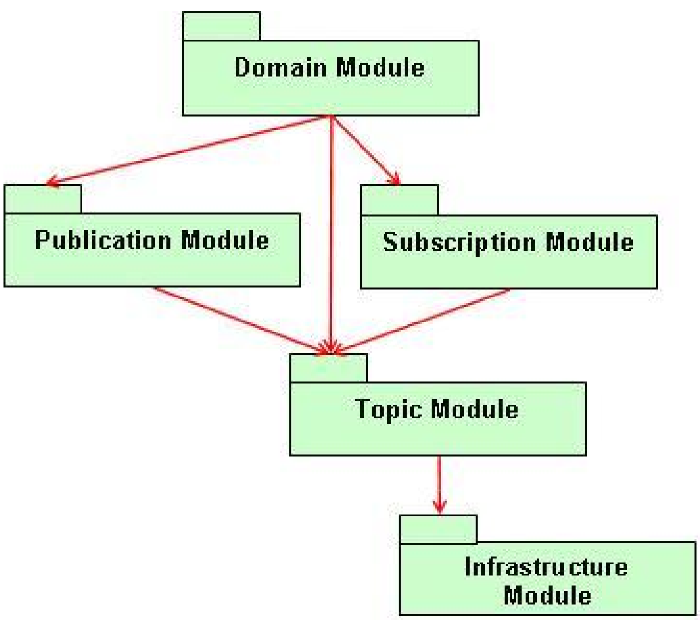
- BP1817: Use the **MANUAL_BY_TOPIC** setting of the **LIVELINESS** **Quality of Service** (QoS) for **Data Distribution Service** (DDS) **Topics** where the presence and health of the **DataWriter** is critical to the proper operation of the system.
- BP1818: Use the **HISTORY** **Quality of Service** (QoS) kind **KEEP_LAST** for **Data Distribution Service** (DDS) **Topics** that represent system state, in that new data-values replace the old values for each Keyed data-object.
- BP1819: Use the **HISTORY** **Quality of Service** (QoS) kind **KEEP_ALL** for **Data Distribution Service** (DDS) **Topics** that represent events or commands where all values written should be delivered to the readers (i.e., new values do not replace old values).
- BP1820: Use **TIME_BASED_FILTER** **Quality of Service** (QoS) to protect **DataReaders** that cannot handle all the traffic that could be written by the writers on that **Data Distribution Service** (DDS)**Topic** and just need periodic updates on the most current data-values.
- BP1821: Use the **Data Distribution Service** (DDS) **LIFESPAN** **Quality of Service** (QoS) to indicate that data is only valid for a finite time period and stale data is discarded after a certain expiration time elapses.
- BP1822: Use the **PARTITION** **Quality of Service** (QoS) to limit the scope of the data written/read on a **Data Distribution Service** (DDS) **Topic** to only the writer/readers that have a common partition.
- BP1823: Use the **Data Distribution Service** (DDS) **RESOURCES_LIMITS** **Quality of Service** (QoS) in platforms with limited memory or in **real-time systems** to properly configure the resources that will be utilized and avoid exhaustion of system resources at run-time.
- BP1824: Use the **USER_DATA** **Quality of Service** (QoS) to communicate metadata on the **DomainParticipant** that may be used to authenticate the application trying to join the Data **Distribution Service** (DDS) **Domain**.
- BP1826: Use the **USER_DATA** **Quality of Service** (QoS) on the **DataWriters** and **DataReaders** to communicate metadata that may provide application-specific information of the entity writing/reading data in a **Data Distribution Service** (DDS) **Domain**.
- BP1828: Use the **Data Distribution Service** (DDS) **OWNERSHIP** **Quality of Service** (QoS) kind set to **SHARED** when each unique data-object within a DDS **Topic** to which multiple **DataWriters** can write.
- BP1829: Use the **Data Distribution Service** (DDS) **OWNERSHIP** **Quality of Service** (QoS) kind set to **EXCLUSIVE** when multiple **DataWriters** cannot write each unique data-object within a DDS **Topic** simultaneously.

P1193: DDS Data-Centric Publish-Subscribe (DCPS)

The **Data-Centric Publish-Subscribe** (**DCPS**) interface is targeted toward the efficient delivery of the proper information to the proper recipients. It provides the application with a **data-centric** information model and is responsible for controlling the lower level layer of the **DDS** infrastructure targeted toward the efficient and reliable delivery of the information to its intended recipients. The DCPS architecture is comprised of five **modules**. The modules build upon each other in a hierarchical inheritance structure. The following table captures the purpose of each of the five modules.

Infrastructure Model	Defines the abstract classes and the interfaces that are refined by the other modules; also provides support for the two interaction styles (notification- and wait- based) within the middleware
Domain Module	Contains the **DomainParticipant** class that acts as an entry point of the Service and acts as a factory for many of the classes; the `DomainParticipant` also acts as a container for the other objects that make up the Service
Topic-Definition Module	Contains the **Topic**, `ContentFilteredTopic`, and `MultiTopic` classes, the `TopicListener` interface, and more generally, all that is needed by the application to define Topic objects and attach **QoS** policies to them
Publication Module	Contains the **Publisher** and **DataWriter** classes as well as the `PublisherListener` and `DataWriterListener` interfaces, and more generally, all that is needed on the publication side
Subscription Module	Contains the **Subscriber**, **DataReader**, `ReadCondition`,and `QueryCondition` classes, as well as the `SubscriberListener` and `DataReaderListener` interfaces, and more generally, all that is needed on the subscription side

The following is a UML Class diagram that represents the five modules and how they relate to each other.



I1199

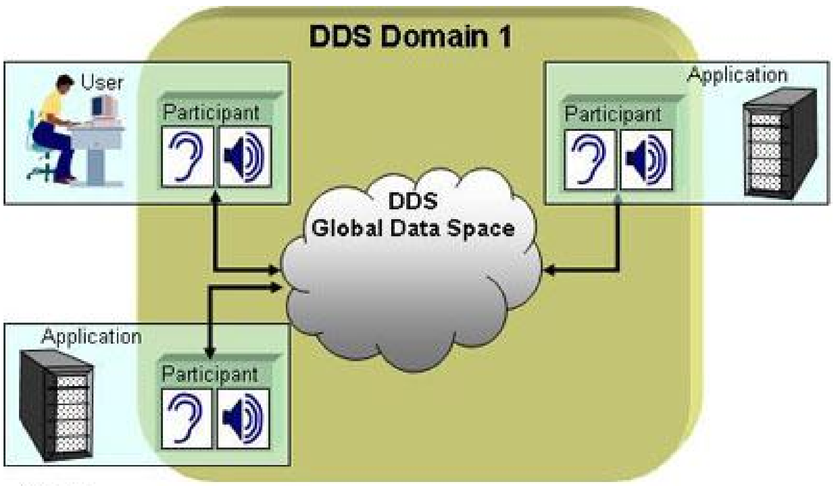
Detailed Perspectives

- DDS Domains - Global Data Spaces [P1194]
- Reading/Writing Objects within a DDS Domain [P1195]

P1194: DDS Domains - Global Data Spaces

DDS allows application developers to create a collection of virtual shared **Global Data Spaces** where separate application processes can share data anonymously. Processes can access (read and/or write) data in the Global Data Space as well as exchange messages on the associated DDS **Domain**.

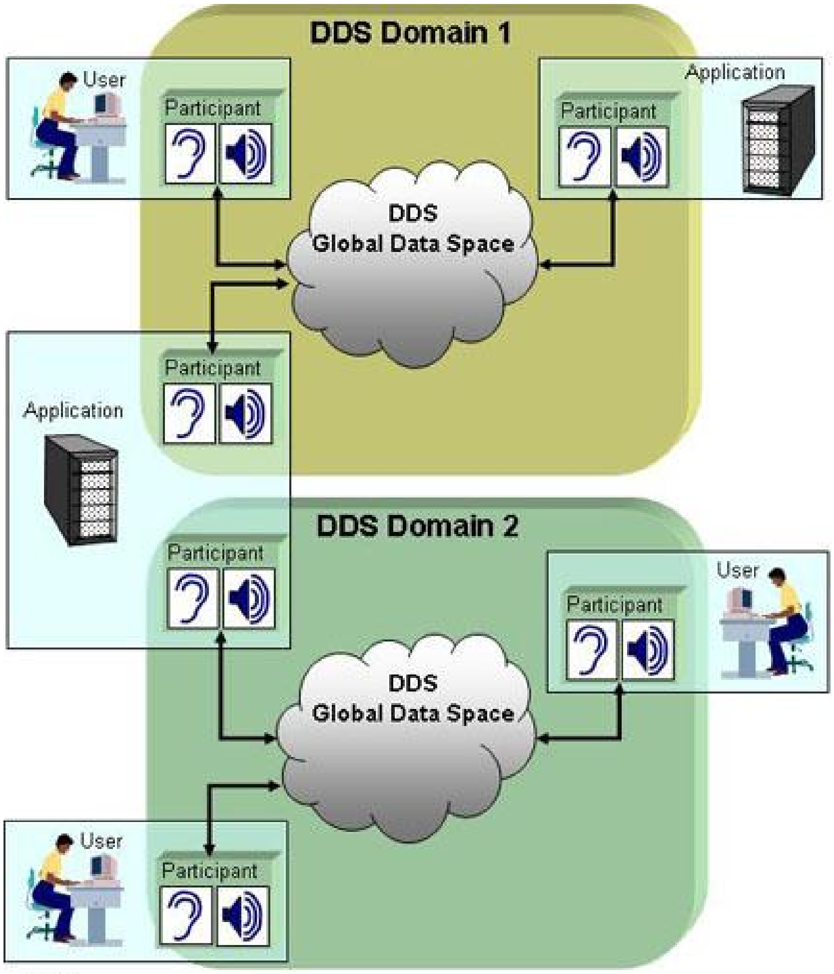
A DDS Global Data Space (called a DDS Domain) is identified by a **`domainId`** that represents an isolated Data Space. The Data Space exchanges no information or messages with other domains. The operating system maintains isolation between DDS Domains by using different port numbers. Each computer process (running on behalf of some user or application) must attach to the desired DDS Domain by creating a DDS **DomainParticipant**. Each **`DomainParticipant`** is owned by the creating process and is only accessible to it.



I1200

> ***Note:*** *The centralized image of a Global Data Space is just a convenient metaphor. In reality the DDS specification mandates that there should be no centralized implementation of the global data and data updates must flow directly from the writer to the readers.*

A distributed system may employ multiple DDS Domains (i.e., Global Data Spaces), each identified by a different **`domainId`**. A single application process may access multiple Global Data Spaces by creating multiple **`DomainParticipants`**, each associated with one of the Global Data Spaces.



I1201

Guidance

- G1770: Explicitly define **Data Distribution Service** (**DDS**) **Domains**.
- G1772: Assign a unique identifier for each **Data-Distribution Service** (**DDS**) **Domain**.

P1195: Reading/Writing Objects within a DDS Domain

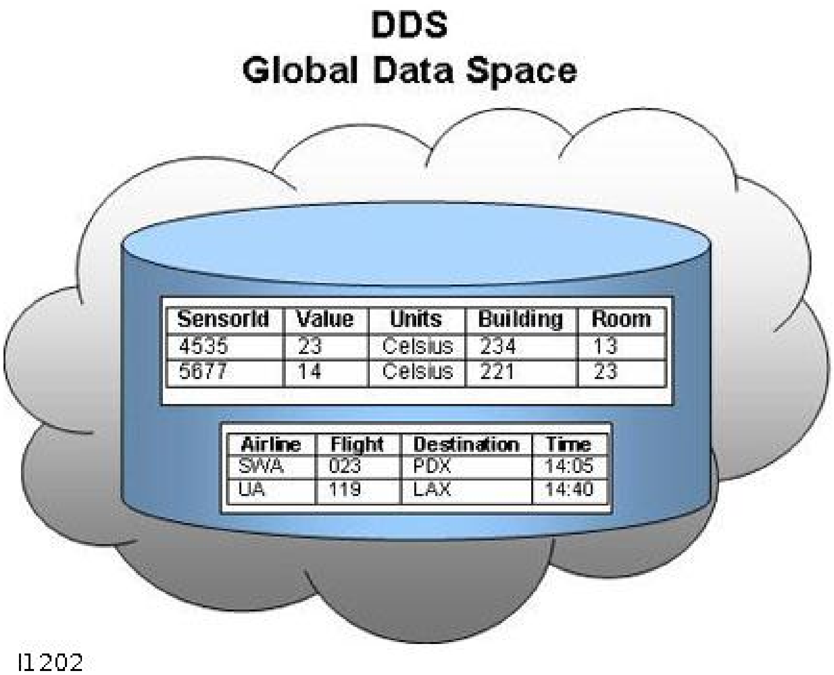
Address the Data Objects in the **Global Data Space** by means of a **Topic** (an application-chosen string that encodes a homogeneous collection of objects) and a ***Key*** (a set of fields inside the data object that uniquely identifies the object within the collection). A **DDS** Topic is an application-chosen string (such as **`Temperature`**) that has an associated schema or format representing the type of the data objects (for example the sensor ID, the value, the units, the location of the sensor, the time-stamp, etc.). The DDS Key is specific to each DDS Topic and uniquely identifies each Data Object within the Topic.

Pictorially one could think of each Topic in the Global Data Space representing a table of related data objects where each row represents the value of an individual data object the columns define the schema (data type of the object), and the key is the column(s) that defines the identity of each object. The table below depicts this concept for the hypothetical **`Temperature`** Topic.

SensorId (Key)	***Value*** : float	***Units*** : string	***Location*** : string	***Timestamp***
4535	23	Celsius	Building 234, Room 13	Tue Oct 31 15:47:42 PST 2006
5677	12	Celsius	Building 121, Furnace 23	Tue Oct 31 15:44:42 PST 2006

Another example is an Airport Information application that defines the Topic **`DepartingFlights`** with a schema consisting of fields containing the following information: Airline, flight number, destination airport, departure terminal, gate, scheduled departure time, expected departure time, and status. In this case the combination of fields Airline and Flight Number provides the Key that uniquely identifies each flight. Updates to the global data space will provide new estimated departure times, departing dates, etc. A display application may read this topic to show all the flights departing in the next three hours.

Airline (Key)	***Flight Number*** (Key)	***Destination***	***Departure Terminal***	***Departure Gate***	***Scheduled Departure***	***Expected Departure***	***Status***
SWA	023	PDX	A	12	10:30	14:05	Departed
UA	119	LAX	A	06	14:27	14:40	Boarding
AS	543	ANC	A	03	14:10	14:20	Boarding
KLM	006	AMS	A	14	14:35	14:35	Boarding
SQ	012	SIN	B	03	15:00	15:20	Go to Gate
JL	001	NRT	B	33	15:45	15:45	Go to Gate
LOT	007	WAW	B	02	16:30	16:30	Wait



Guidance

- G1141: Base **data models** on existing data models developed by **Communities of Interest** (**COI**).
- G1146: Include information in the **data model** necessary to generate a **data dictionary**.
- G1147: Use **domain analysis** to define the constraints on input data validation.
- G1148: **Normalize** data models.
- G1810: Use **data models** to document the data contained within the **Data Distribution Service** (**DDS**) **Data-Centric Publish Subscribe** (**DCPS**).

Best Practices

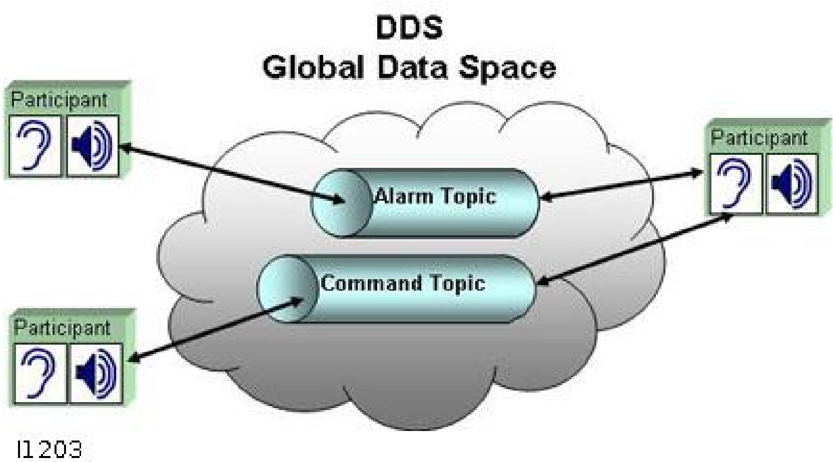
- BP1145: Use vendor-neutral **conceptual/logical models**.
- BP1254: For **command-and-control** systems, use the names defined in the Joint Command, Control and Consultation Information Exchange Data Model (JC3IEDM) for data exposed to the outside communities.
- BP1397: Identify and develop use cases or reuse existing use cases as appropriate as early in the data engineering process as possible to support **data model** development.

P1196: Messaging within a DDS Domain

A **DDS Topic** acts like a virtual message-queue or pipe when DDS is used for messaging. Writers send messages though the Topic and readers access messages using the same Topic.

Topics for DDS messages are bound to an application-defined schema in advance; for example, an ***Alarm*** message where the schema consists of source identifier, the kind of alarm, the location, a time-stamp, and the urgency level. **DomainParticipants** can publish and subscribe messages by specifying the Topic and the associated contents.

The Topics used for messaging also live within a DDS **Domain** (i.e., **Global Data Space**) identified by a unique **`DomainId`**. Similar to the data-object paradigm, the middleware keeps the messaging Topics separated within different DDS Domains by using different port numbers.



> ***Note:*** *The centralized image of a pipe is only a convenient concept. In reality, the DDS specification mandates that there should be no centralized implementation of a pipe in DDS. Messages must flow directly from the sender to the receivers.*

The distinction between reading/writing data and receiving/sending messages is essentially a property of the Topic. Some Topics represent data (if the identify certain fields as Keys) and others represent messages (if they do not contain specify Keys). In addition, use different **Quality of Service** settings to attain the proper semantics. For example, associate Topics representing data with a **`HISTORY`** QoS setting of **`KEEP_LAST`** whereas Messages typically use a **`HISTORY`** setting of **`KEEP_ALL`**.

> ***Note:*** *For more details on this subject please refer to the introductory material on DDS available at the OMG DDS Portal.*

Guidance

- G1796: Explicitly define **Data Distribution Service** (**DDS**) **Domain Topics**.
- G1798: Explicitly define all the **Data Distribution Service** (**DDS**) **Domain data types**.
- G1799: Explicitly associate data types to the **Data Distribution Service** (**DDS**) **Topics** within a DDS **Domain**
- G1800: Explicitly identify Keys within the **Data Distribution Service** (**DDS**) **data type** that uniquely identify an instance of a data object.
- G1801: Explicitly define a **Topic Quality of Service** (**QoS**) for each **Data Distribution Service** (**DDS**) Topic within a DDS **Domain**.

Part 5: Developer Guidance > Middleware > Messaging > Data Distribution Service (DDS) > DDS Data Local Reconstruction Layer (DLRL)

P1197: DDS Data Local Reconstruction Layer (DLRL)

The **Data Local Reconstruction Layer** (**DLRL**) is an optional part of the **Data-Distribution Service** (**DDS**) specification that provides a local ***object-cache*** abstraction built upon the core **DCPS** layer and requires application objects to comply with the DLRL object metamodel which includes collections and relationships.

> ***Note:*** *The DLRL, a recent addition to the DDS specification, is particularly rich; implementations using this upper-level profile of the specification are emerging.*

Application developers use the DLRL to do the following:

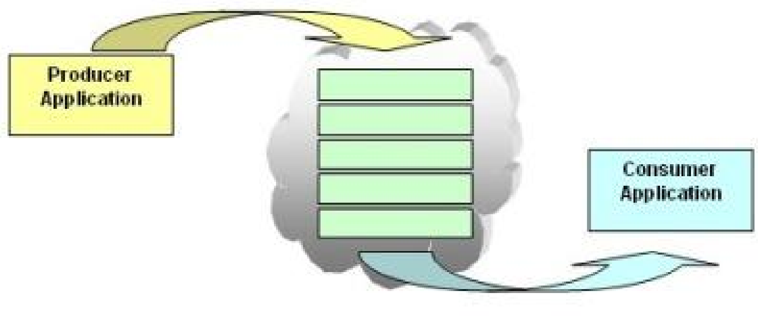
- Describe classes of objects with the associated methods, data fields and relations
- Attach data fields to **Data-Centric Publish-Subscribe** (**DCPS**) entities
- Use native language constructs to manipulate objects (i.e., create, read, update, delete) using native language constructs to seamlessly interact with the DCPS layer
- Manage objects and pointers to objects in a cache

Best Practices

- BP1832: Handle all **Data Distribution Service** (DDS) **Data Local Reconstruction Layer** (DLRL) Exceptions.
- BP1833: Use the **Data Distribution Service** (DDS) Object Model Profile for accessing message data as objects.

P1048: Messaging with MSMQ

Messaging in **.NET** uses Microsoft Message Queue (**MSMQ**). MSMQ is responsible for reliably delivering **messages** between applications inside and outside the enterprise. MSMQ ensures reliable delivery by placing messages that fail to reach their intended destination in a queue and then resending them once the destination is reachable.



I1067

MSMQ also supports transactions. It permits multiple operations on multiple queues, with all of the operations wrapped in a single transaction, thus ensuring that either all or none of the operations will take effect. Microsoft Distributed Transaction Coordinator (MSDTC) supports transactional access to MSMQ and other resources.

Best Practices

- BP1111: Mark all **Microsoft Message Queue** (**MSMQ**) messages as recoverable.
- BP1112: Specify all **Microsoft Message Queue** (**MSMQ**) queues as transactional if they support multiple-step processes.
- BP1227: Do not allow installation of **MSMQ**-dependent clients.
- BP1230: Do not use the **MSMQ** **`SupportLocalAccountsOrNT4`** feature.

P1078: Web Services

A **Web service** is an application that exists in a distributed environment, such as the **Internet**. A Web service accepts a request, performs its function based on the request, and returns a response. The request and the response can be part of the same operation, or they can occur separately in which case the consumer does not need to wait for a response. Web services tend to fall into one of two camps: those that use **Extensible Markup Language** (**XML**) messages that follow the **SOAP** standard, popular with traditional enterprises, and **Representational State Transfer** (**REST**) based communications. SOAP Web services usually have a formal interface described in a machine-processable format (specifically, **Web Services Description Language** or **WSDL**). REST Web services do not require XML, SOAP, or WSDL service-**API** definitions but best practice recommends using standardized formats and protocols.

A Web service can reside on top of existing legacy applications and expose services to the net. The Web services architecture illustrated below implements the **service-oriented architecture** pattern. For more information on design patterns, see *Web Service Patterns: Java Edition* by Paul B. Monday (http://apress.com/book/view/9781590590843).

Web Service Models

Web services have traditionally been used to connect people to **services**. However, as the Web service infrastructure has matured, a new model has emerged, the service-to-service model.

Traditional Model

In a classic Web service, a request is usually made to a Web service using a **Web browser**. The request is submitted to the Web service using **HTTP** or **HTTPS** over the **Internet** or an **intranet**. The Web service processes the request and returns an **HTML** page that can be displayed in a Web browser.

A classic Web service has the following characteristics:

- **Web pages** appear via a Web browser
- Connection is via **TCP/IP**
- Transport is HTTP/HTTPS
- Message format is HTML

Service-to-Service Model

Application servers used to be responsible for providing machine-to-machine services. Now **Web servers** can handle similar work. The Web server can pass a request as an **XML** payload embedded in a TCP/IP and HTTP request, process the data, and respond. The response is typically in the form of an HTML Web page or an XML payload that a **client** application can use.

Machine-to-machine Web services have the following characteristics:

- Two independent applications
- Two independent **servers**
- Connection is via TCP/IP
- Transport is HTTP (port 80)
- Message format is XML payload in **SOAP** format

Key Characteristics

Some key characteristics of Web services include the following:

- High-overhead interactions; may be too heavy for some applications
- **Loosely coupled** collaborators (e.g., client/server)
- Multiple layers of **parsing**, **marshalling**, and un-marshalling
- Non-standard content

- Standard interaction **protocol**
- No support for **services** such as **messaging** and security
- Infant technology
- No support for pass-by-reference

Detailed Perspectives

- SOAP [P1068]
- Web Services Compliance [P1081]
- REST [P1398]
- WSDL [P1082]
- Insulation and Structure [P1035]
- Universal Description, Discovery, and Integration (UDDI) [P1075]
- Service Definition Framework [P1296]

Guidance

- G1087: Validate all **Web Services Definition Language** (**WSDL**) files that describe **Web services**.
- G1088: Use isolation **design patterns** to define system functionality that manipulates **Web services**.
- G1090: Do not **hard-code** a **Web service's endpoint**.

P1068: SOAP

SOAP is an **XML** message-based **protocol**. SOAP is lighter weight and requires less programming than similar protocols such as **CORBA** and **Distributed Component Object Model** (**DCOM**). SOAP defines an extensible messaging framework independent of programming models and other implementation-specific semantics.

The **World Wide Web Consortium** (**W3C**) provides this description of SOAP:

> ***Note:*** *Prior to SOAP v1.2 the official name was the Simple Object Access Protocol (SOAP); W3C dropped the acronym expansion in SOAP v1.2.*

"SOAP Version 1.2 (SOAP) is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics." [R1002]

Two major design goals for SOAP are simplicity and extensibility. SOAP attempts to meet these goals by omitting distributed-system features from the messaging framework. Such features include but are not limited to reliability, security, correlation, routing, and Message Exchange Patterns (MEPs). While it is anticipated that many features will be defined, this specification provides specifics only for two MEPs. Other features are left to be defined as extensions by other specifications.

SOAP is a protocol for exchanging structured information in a decentralized, distributed environment. It consists of three parts that support interoperability:

- a framework or envelope that describes what is in a message and how to process it
- a set of encoding rules for the application-defined **data types** used in the message
- a convention for representing **remote procedure calls** and responses that allow applications to correlate requests and responses

Key Characteristics

SOAP is an XML message-based **wire protocol**.

SOAP is implemented by many language bindings.

SOAP is inherently stateless; consumers of SOAP services manage their own state.

SOAP relies on other standards to implement security directly.

Message Styles

The W3C **WSDL** 1.1 Specification identifies two message styles: Document and RPC. The purpose of the styles determines how the content of the SOAP message body is formatted.

Document	The SOAP Body contains one or more child elements called parts. There are no SOAP formatting rules for what the SOAP Body contains; it contains whatever the sender and the receiver agree upon. ***Note:*** *There is a Wrapped form of this style that is required to interoperate with Microsoft **Web services** using Document style. There is no specification that defines this style.*
RPC	**RPC** implies that the SOAP Body contains an element with the name of the method or remote procedure being invoked. This element in turn contains an element for each parameter of that procedure.

Note: *Document style can be interpreted as either an* ***XML*** *string or as a W3C* ***Document Object Model*** *(****DOM****) Document Element. Microsoft has a technique called Wrapped that encapsulates the information being exchanged, regardless of the style.*

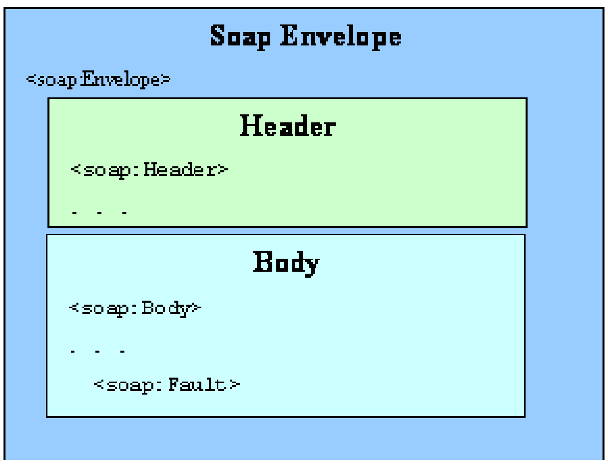
Serialization Formats

For applications that use **serialization**/**deserialization** to abstract away the data wire format, there is one more choice to be made: the serialization format. The following table describes the two most popular serialization formats today.

SOAP Encoding	SOAP encoding uses a set of rules to serialize the data transferred between the **client** and the **server**. The rules are defined in section 5 of the **WSDL** 1.1 Specification. These rules are also referred to as "section 5 encoding." The rules specify how to serialize objects, structures, arrays, and object graphs and directly use the predefined **XML Schema** data types. Generally, an application using SOAP encoding should use the **RPC mssage** style.
Literal	Data is serialized according to an independent external schema. There are no preset rules for serializing objects, structures, and graphics, etc., in the literal encoding style. The industry is overwhelmingly embracing XML Schemas.

Structure

A SOAP message comprises three parts: an envelope, an optional header, and a required body. The envelope encapsulates the other two elements. The optional header contains one or more header elements that contain meta-information about the method calls.



I1046

Envelope	The Envelope is the root of the SOAP request. At a minimum, it defines the SOAP namespace for SOAP 1.2. The envelope may define additional namespaces.
Header	The Header contains auxiliary information as SOAP blocks, such as authentication, routing information, or transaction identifier. The header is optional.
Body	The Body contains the main information in one or more SOAP blocks; for example, a SOAP block for RPC call. The body is mandatory and it must appear after the header.
Fault	The Fault is a special block that indicates a protocol-level error. If present, it must appear within a Body element.

SOAP is a protocol for exchanging structured information in a decentralized, distributed environment. It consists of three parts that support interoperability:

- a framework or envelope that describes what is in a message and how to process it
- a set of encoding rules for the application-defined datatypes used in the message
- a convention for representing remote procedure calls and responses that allow applications to correlate requests and responses

Guidance

- G1082: Use the document-literal style for all data transferred using **SOAP** where the document uses the **World Wide Web Consortium** (**W3C**) **Document Object Model** (**DOM**).
- G1088: Use isolation **design patterns** to define system functionality that manipulates **Web services**.
- G1093: Implement exception handlers for **SOAP**-based **Web services**.
- G1095: Use **W3C** fault codes for all **SOAP** faults.

P1081: Web Services Compliance

The **Web Services Interoperability Organization** (**WS-I**) is an open industry effort to promote **Web services** interoperability across platforms, applications, and programming languages.

The WS-I goal is to be a standards integrator to help Web services advance in a structured, coherent manner as standards evolve independently and in parallel. To support this, WS-I is developing a set of profiles that provide implementation guidelines for how to use related Web services specifications together for best interoperability.

WS-I finalized the ***Simple SOAP Binding Profile*** as of 24 August 2004, the ***Attachments Profile*** as of 20 April 2006 with an errata dated 1 March 2008, and the ***Basic Profile 1.1*** as of 10 April 2006. WS-I is also developing Sample Applications, Testing Tools and an XML Schema Work Plan.

Guidance

- G1080: Adhere to the **Web Services Interoperability Organization** (**WS-I**) Basic Profile specification for **Web service** environments.
- G1082: Use the document-literal style for all data transferred using **SOAP** where the document uses the **World Wide Web Consortium** (**W3C**) **Document Object Model** (**DOM**).
- G1083: Do not pass **Web Services-Interoperability Organization** (**WS-I**) **Document Object Model** (**DOM**) documents as strings.

P1398: REST

The **Representational State Transfer** (**REST**) architectural style is resource-centric service-oriented approach for performing simple Create/Read/Update/Delete (CRUD) operations on remote information. REST consists of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Unlike **SOAP**, REST responses are built around the transfer of context ***representations*** of whole ***resources***. A resource essentially can be any coherent and meaningful collection of data that may be addressed. A representation of a resource typically is a document that captures the current or intended state of a resource.

A number of different protocol bindings can be the basis of RESTful architectures. Typically, resources are formatted in **Extensible Markup Language** (**XML**) or JavaScript Object Notation (JSON), but other Multi-Purpose Internet Mail Extensions (MIME) types may be used. Likewise, the typical Transport is the **Hypertext Transfer Protocol** (**HTTP**), but the Extensible Messaging and Presence Protocol (XMPP), **Java Message Service** (**JMS**) and **Simple Mail Transfer Protocol** (**SMTP**) have also been used. REST is not a standard; it is a way of using other application layer protocol standards that already provide a vocabulary for applications based on the transfer of meaningful representational state. REST is simpler to use than SOAP, which requires writing or using a provided middleware for both the server and the client.

A RESTful service (also called a RESTful service **API**) is a simple service implemented using a MIME data encoding, a Transport, and the principles of REST. It is a collection of resources, with three defined aspects:

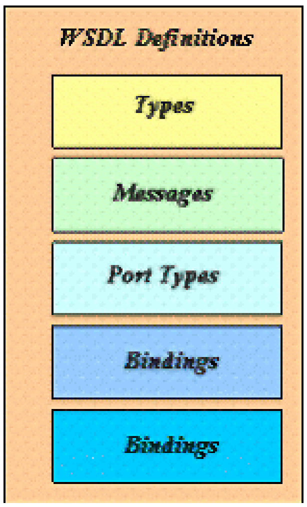
- the base **Uniform Resource Identifier** (**URI**) for the service
- the MIME type of the data supported by the service
- the set of operations supported by the service using the transport protocol's methods (e.g., HTTP **`POST`**, **`GET`**, **`PUT`** or **`DELETE`**)

P1082: WSDL

Web Services Description Language (**WSDL**) is an **XML**-based language that is used to describe a **Web service**. It describes the operations that are available from the Web service and it describes the data that flows between the consumer and the producer of the service. In addition, it describes the **endpoint** that locates the Web service.

An endpoint is a connector construct used in assembling a service, system, Node or enterprise from components. Specific endpoints represent and label one side of an interface used to exchange information with partner endpoints on other components. Endpoints bind a component's internal application data and processes to infrastructure resources at the interface. In the case of Web services, bindings are to a network protocol, its operations and message-formatted data. Network infrastructure Transport endpoints are called ports.

Related endpoints connect components into services bound to, and running on top of, infrastructure or middleware resources. This enables the reuse of standardized bindings and endpoints (port types) and considerably eases interoperability.



I1060: WSDL Definitions

WSDL uses XML to define several types of standardized web services endpoints and bindings. Currently these types include document-oriented and procedure-oriented. WSDL is extensible in that an architect or designer chooses the most appropriate binding and port and the associated message format and network protocol the service's endpoints and application messages are to use.

Guidance

- G1085: Establish a **registered namespace** in the **XML Gallery** in the **DoD Metadata Registry** for all DoD Programs.
- G1087: Validate all **Web Services Definition Language** (**WSDL**) files that describe **Web services**.

P1035: Insulation and Structure

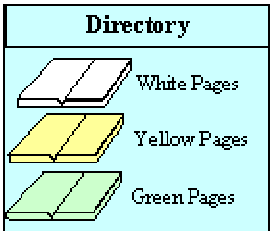
Insulating the user of **Web services** from the implementation of the services enhances the maintainability and portability of the overall system and aids in the migration to net-centricity. Application developers can use the facade or adapter design pattern for Web services to insulate applications from the implementation details of the service. Services can then change over time to match changing requirements and deployments. Legacy functionality can be similarly wrapped via a service. It is important to not directly expose vendor-specific functionality via the services interface to enable the ready reimplementation of the service if necessary.

Guidance

- G1087: Validate all **Web Services Definition Language** (**WSDL**) files that describe **Web services**.
- G1088: Use isolation **design patterns** to define system functionality that manipulates **Web services**.
- G1090: Do not **hard-code** a **Web service's endpoint**.
- G1237: Do not **hard-code** the configuration data of a **Web service** vendor.

P1075: Universal Description, Discovery, and Integration (UDDI)

The **Universal Description, Discovery, and Integration** (**UDDI**) standard is an industry initiative for a **Web services** registry. It enables businesses to access a universal pool of Web services. The UDDI registry contains yellow pages, white pages, and so-called "green pages," like a phone book.



I1062

White pages	List point of contact information, such as • Name • Address • Phone • Fax • email
Yellow pages	List services that are available from businesses, such as • Weather data • Software development • Project management
Green pages	List service properties, such as • Business processes • Service descriptions • Binding information • Categorization of services • XML version, type of encryption, and Document Type Definition (DTD)

UDDI is a platform-independent, open framework that allows automated consumers and suppliers to find each other, assess mutual compatibilities, negotiate terms, and build the relationship. It supports human interaction as well as machine-to-machine communication. People can use a UDDI browser to review services and find point-of-contact information (white pages), and business information (yellow pages).

Like the **Domain Name System** (**DNS**), the UDDI registry comprises a network of **servers** on the internet. It is a **SOAP**-based mechanism. The **API** specification focuses on the storage, organization, and architecture of the registry.

The UDDI project takes advantage of **World Wide Web Consortium** (**W3C**) and **Internet Engineering Task Force** (**IETF**) standards such as **eXtensible Markup Language** (**XML**) and **HTTP** and Domain Name System (DNS) **protocols**.

Guidance

- G1127: Use a **UDDI** specification that supports publishing discovery services.
- G1131: Use standards-based **Universal Description, Discovery, and Integration** (**UDDI**) **application programming interfaces** (**APIs**) for all UDDI inquiries.

P1296: Service Definition Framework

A Service Definition Framework (SDF) provides a common frame of reference for service users, customers, developers, providers, and managers. Its structure and methodology enable full definition of the Service Access Points (SAPs) for a service. The purpose of the SDF is not to describe the internal workings of a service. Rather, it concentrates on defining the boundary conditions for accessing a service through its service access point. The SDF also includes specific technical parameters and engineering-level data that prospective service developers and providers can use to design and implement new enterprise service offerings.

Complete an SDF entry for each enterprise service. Subsequently, register each service in a service registry (e.g., the NCES Service Discovery service or the Air Force Service Management Tool). The SDF provides the basis for a design specification where potential implementers of a new service will find the information required to implement the service. The SDF should address the following information for each service:

- What the service does
- How the service works (from a black box perspective)
- Any required security mechanisms or restrictions
- Any pertinent performance or **quality of service** (**QoS**) information
- Points of contact for the service:
  - Who is providing the service
  - Who is responsible for the daily operation of the service
  - Who is developing the service
- The specifics of how to bind to (access or use) the service.

Service Profiles

A service profile captures the black box architecture of a service. It would precede and guide one or more service implementations documented in association with the SDF. The use of a service profile becomes critical in the case of those enterprise services that have more than one implementation and implementer across the enterprise. The profile provides the guidance needed to ensure that multiple service implementations provide a common consumer interface and are interoperable.

Proposed SDF Lifecycle

The proposed SDF lifecycle is to assist service implementers in developing and maintaining an SDF entry during the lifecycle of an enterprise service. Scenarios include the following:

- Creating an SDF Entry
- Changing a Registered SDF Entry
- Deprecating a Registered SDF Entry
- Accessing a Registered SDF Entry

The proposed SDF Lifecycle is consistent with the DoD Acquisition Steps defined in the DoD 5000 series Directives and Instructions. The table below describes the proposed steps for the SDF lifecycle, along with associated business processes, the service owner and mandatory categories for each phase.

Lifecycle Element	***Description***	***Business Processes***	***Service Owner***	***Mandatory Categories by Phase***
Concept Development	Identify possible need for a new service and create	Examine mission threads and search for services to fulfill them. Identify capability gaps. These gaps become services within	Portfolio Manager	Service name, service description, schedule

	justification for service	classification domains. Create high level business or mission capability statement. Perform initial cost analysis and Analysis of Alternatives. Define acquisition approach and organizations to execute following phase		
Requirements and Architecture	Define service architecture and requirements	Identify specific organizations for each type of user, Define service requirements and semantics. Define service architecture to include interaction with other services and systems, basic service capabilities and service deployment approach. Perform Systems Program Office (SPO) level cost analysis.	Portfolio Manager to Acquirer	Semantic model, pedigree, information security marking, cpoints of contacts
Service Design	Create service "black box" interface specs for handoff to developers	Start configuration management: • finalize semantics • point to metadata repository • finalize classification details • determine service level agreements (SLAs) offered, finish WSDL	Acquirer	Operations, number of operations, security mechanisms, access criteria and restrictions, service level specification, network requirements, SAP
Service Build	Develop/purchase service	Development (generally follows contractor's best practices)	Acquirer	Consumer patterns, schedule Beta, operational reference
Service Testing	Assure service meets specifications and requirements	Acceptance test: • meets specifications • plays well with others • interoperability "seals of approval" from authoritative bodies	Acquirer to Operator/ Sustainer	Schedule: integration
Service Deployment	Install service instance(s)	Configuration management: • updating humans/summary from monitoring • measuring coarse-grained triggers for action (scaling)	Operator/ Sustainer	Schedule: deployment
Service Operation	Operate service; concludes with EOL announcement.	Configuration management: • updating humans/summary from monitoring	Operator/ Sustainer	Schedule: operation

		• measuring coarse grained triggers for action (scaling)		
Service Deprecation	Service is still being operated but is to be replaced or retired; concludes with service EOL	Work with consumers to adopt new version of service, or replacement service(s) as appropriate	Operator/ Sustainer	Schedule: deprecation
Service Retired	Service is not operating; service definition information is still available for use/ reuse; concludes with purging of service definition information	Service migration and reuse	Sustainer	Schedule: retire

Notional SDF Concept of Operations

The Notional SDF Concept of Operations (CONOPS) outlines a theoretical concept for Service Discovery. The SDF concept focuses on why a service is needed and how it is used. The Notional SDF CONOPS addresses the following issues:

- Key Assumptions:
  - Location, composition, extensibility, syntax, failover, information assurance, alignment to COIs and applicable security classification level
  - Governance
  - Services are made available via an Enterprise Service Bus or via the Web services stack
  - The SDF will be used for defining services from many sources and multiple languages
- Creation of an SDF Entry
  - Two scenarios in which a service will require the creation of an SDF entry:
    - Capability already exists and will be "service enabled"
    - Capability does not exist
  - The SDF entry becomes part of the Key Interface Profile (KIP) for that service
- Services Lifecycle and SDF Development Process Flow
  - Establishment of a business case
    - Warfighter or COI has defined a need
    - Service requirements analysis and definition
    - Funding
    - Resources assigned
  - Design
  - Development
  - Test
  - Deploy
- SDF Implementation
  - SOA

    - Publishing
    - Discovery
    - Binding
  - Operations and maintenance
    - Change Management
    - Deprecation
    - Monitoring and maintenance

  Under SDF Implementation, NESI also advises that ConOps include Portfolio Management and Capability Planning. NESI will add these components in future versions.

SDF Considerations

- Describe all services using a standard Service Definition Framework (SDF).
  - Adhere to DoD Policy as a core definition for the SDF
  - Extensions can be made to core definition to suit specific needs
- May want to extend "Required" fields (from core SDF)
- Capture and track associated Lifecycle Phase
- The "Owner" of the service (and SDF) will change as the Lifecycle Phase changes; update the SDF at each Lifecycle phase.
- Begin capturing SDF data at the earliest possible Lifecycle Phase, preferably Concept Development.
  - Not all information will be available
  - Recommended to trace service capability back to operational needs, shortfalls and requirements
- Make SDF data accessible by storing contents either in an XML document in conformance with the XML Schema or in the form of a set of database tables with a front-end.
  - The XML Schema or database tables will contain all elements and attributes of the core (and extended) SDF
  - Common practices for database tables with a front-end include the following:
    - Group SDF data elements into logical categories and reflect such in the User Interface (UI) for ease of use; do not just provide one large input form
    - Reports are high value; being able to view SDF data via reports allows for relationships to be discovered and services to be managed (Portfolio Management, Capability Based Planning)
    - Role-based access for data editing is vital for information assurance and integrity; don't want Service Owner A to edit Service Owner B's SDF
    - Enforce security policies at the Data Level rather than at the application and/or UI level; provides stronger information assurance and accountability (audits); allows data entries and data fields to be customized to each user/role
- Capture SDF data from discrete choices (lists) rather than just "free text"; while free text can be searched via key word, it does not allow as much capability for data relationships and data mining.
- Make SDF data understandable and use terminology/labels relevant to the particular domain (enterprise).
- Designate minimally required data with respect to appropriate Lifecycle Phase needed for a complete understanding of the service at that phase.
  - Tie "Required" fields to lifecycle phases; some information may not be available at earlier phases, but would be required before eventually moving into a later phase.

SDF Template

The SDF Template provides a sample logical model to help the service implementer to understand the big picture for the Service Definition Framework. The logical SDF model, summarized in the following table, provides the primary service element categories and service element names. Each service element represents information that may or may not be relevant to the particular service being described. Some service elements may only be applicable during certain phases in the service lifecycle. Other service elements may not apply to specific technologies.

The attributes of a service that are necessary to effectively define and describe the service are identified within the SDF and organized into the following categories:

- Interface information
- Security information
- Service level information
- Implementation information
- Point of contract (POC) information
- Service Access Point (SAP) information

All categories, with the exception of the SAP, are abstract and allow defining the service so as to encourage semantic understanding of the service. The last category (SAP) is the concrete portion that is filled in after the service implementation and deployment. The SAP binds the abstract service specification to the concrete service interface as implemented by an actual process. Specific syntax, protocols and IP address required to use the functionality provided by the service are contained in the SAP.

In the table, the service elements have an associated cardinality for inclusion in the SDF. Cardinality is interpreted as follows:

- Cardinality = 1: Element is mandatory, one instance only
- Cardinality = 1..n: Element is mandatory, one to many ("n" = no upper limit, or upper limit is specified)
- Cardinality = 0..1: Element is optional, but limited to one instance if it is present
- Cardinality = 0..n: Element is optional, and there may be one instance or more if it is present.

Table 2 has an additional column, which is the recommended lifecycle phase where the given service element applies. A detailed specification of Service "Data" Elements will be included in a future release of NESI.

Service Category Element	***Service Element***	***Cardinality***	***Service Development Lifecycle Phase***
Interface information	ServiceName	1	Concept Development
	Service Description	1	Concept Development
	Semantic Model	0..1	Requirements & Architecture
	NumberOfDataTypes	1	Service Design
	DataTypes	0..n	Service Design
	NumberOfOperations	1	Service Design
	Operations	1..n	Service Design
	ServicePedigree	1	Requirements & Architecture

Security information	SecurityMechanisms	1	Service Design
	AccessCriteriaAndRestrictions	1	Service Design
	InformationSecurityMarking	1	Requirements & Architecture
Service level information	NumberOfServiceLevels	1	Service Design
	ServiceLevelSpecifications	0..n	Service Design
	NetworkRequirements	0..1	Service Design
Implementation information	ConsumerPatterns	0..1	Service Build
	NumberOfScheduleDates	1	Concept Development
	Schedule	1..n	Concept Development
	NumberOfOperationalReferences	1	Service Build
	OperationalReference	0..n	Service Build
	VersioningApproach	0..n	Service Design
POC information	NumberOfContacts	1	Requirements & Architecture
	Contacts	1..n	Requirements & Architecture
SAP information	NumberOfSAPs	1	Service Design
	ServiceAccessPoint	0..n	Service Design

P1389: Enterprise Service Bus (ESB)

There are differing definitions within the computing industry and academia for the term **Enterprise Service Bus** (**ESB**). Some definitions describe an ESB as an **architectural style** or enterprise **design pattern** and other definitions describe an ESB as a middleware layer provided by a product or collection of products.

This perspective does not provide a new definition of ESB; rather, it explains ESB as an architectural style that provides distributed invocation, **mediation**, and end-to-end management and security of services and service interactions to support the larger architectural style known as **Service-Oriented Architecture** (**SOA**). In this perspective, as well as throughout NESI, the terms ***ESB*** and ***ESB architectural style*** are synonymous.

A common goal for implementing an ESB is to reduce **coupling** in service interactions by providing architectural components which act as intermediaries to provide mediation and service virtualization. This reduced coupling provides for a clean separation of concerns in areas such as implementation technologies and standards, transport protocols, design and messaging patterns, configuration management, personnel (to include developers, administrators, and operational support personnel), and organizations.

> ***Note:*** *This definition of an ESB as an architectural style does not preclude vendors from providing solutions that implement the ESB architectural style, nor does it prevent one from calling an ESB implementation an Enterprise Service Bus.*

The ESB architectural style requires the hosting of services. Without services, the resulting architecture would be nothing more than **Message Oriented Middleware** (**MOM**) or a message broker. Implementing these services does not necessarily requires the use of SOAP; the ESB architectural style often exposes many types of service implementations such as services based on **Representational State Transfer** (**REST**; see also the REST [P1398] perspective in NESI Part 5) or **Java Message Service** (**JMS**).

The ESB architectural style leverages the concept of a bus as a subsystem that transfers data between endpoints. Traditionally, without the use of an ESB, the service provider and the consumer engaged in an interaction must agree on the same protocol and message format. In essence, each protocol and message format becomes its own bus.

In contrast, an ESB implementation behaves as a universal bus by providing adapters that allow service providers and service consumers to interact without concern for the specific protocol and format of each other. The end result is that the provider and consumer are less coupled (for example in protocol, location, and message format). Each is still coupled to an underlying protocol and format that are usually based on open standards. For example, a service consumer that wants a service delivered using HTTP can easily interact with a service provider that offers services using JMS.

An ESB generally has core characteristics in the areas of services, invocation, messaging, mediation, transport, management, and security as shown in the table below.

Services	Support to host and manage services
Invocation	Support for consumers to locating and binding to services
Messaging	Support for service providers and consumers to communicate through the exchange of well-defined messages through various communication patterns to include synchronous, asynchronous, and publish and subscribe
Mediation	Support for transformation, aggregation, adaptation, orchestration, and choreography. Mediation may occur on many areas to include message content, transport protocol, **quality of service** (**QoS**) parameters, service version, etc.
Transport	Provides for routing, transport, security, and guaranteed delivery of message between service providers and service consumers, often through the use of message routers and adapters for various standards based communication protocols

Management	Support for the management of service interactions and status to include, alerting, auditing, logging, QoS monitoring, configuration management, and metric collection
Security	Support for enforcing enterprise security polices and adapting to security threats

In addition to these core characteristics, an ESB generally provides the following capabilities:

- An ESB allows for the service providers to provide data at a rate independent from the consumer's consumption rate. ESB implementations often supports the pairing of consumer and providers based on QoS parameters and by providing message filtering capabilities.
- An ESB provided an opportunity for service providers to compartmentalize their implementations behind a well-defined interface so that consumers can use the service without having to understand the internal details of the service.
- An ESB enables loose coupling of service providers and consumers which aids integration and composeability. Service consumers are blind to implementation technologies used by service providers and vice versa. Any number of service providers may process a request message dynamically based on QoS or location. An ESB provides support for late binding of service endpoints. Consumers and providers do not have to agree on transport protocol or endpoint addresses.
- An ESB support service versioning by isolating changes to services. Service consumers can continue making request to older versions of a service while an ESB provides mediation services.
- An ESB reduces the number of point-to-point contacts between service providers and service consumers easing integration and making impact analysis for changes or vulnerabilities easier.
- An ESB provides service logging to include what services are used, who uses them, the performance of the service interactions, and exceptional conditions and errors.
- An ESB supports fault tolerance through concepts such as intelligent routing, redundant service providers, and execution of a formally specified business process to support and implement the recovery process.
- An ESB supports composition and execution services to support business processes to include long-running transactions. This is usually done through the use of a formally specified business process.
- ESB implementations are aided by existing developer and engineer skills with technologies such as **XML**, XML Path Language (XPATH), and eXtensible Style Language Transformations (XSLT).
- An ESB is an enabler for reuse by allowing for expose legacy systems through the use of adapters resulting in a possible cost savings.
- An ESB helps manage risk through incremental SOA implementation.
- An ESB Supports distributed SOA implementation.

Although an ESB may provide many advantages for SOA implementation, several challenges remain:

- There is not an industry-wide agreed upon definition for ESB and there is not a single ESB standard. As a result, vendors support various capabilities within their ESB support products which can lead to vendor dependence and coupling.
- An ESB infrastructure may increase latency between service consumers and service providers compared to a direct stovepipe connection.
- An ESB infrastructure can become a major point of failure in a system as well as a major target for penetration of denial of service attacks.
- Mapping between information exchange patterns may not be optimal.

The following general guidelines, in addition to formal NESI guidance, may help to mitigate these concerns.

- Content providers should be responsible for translations, not the ESB since it forces the ESB development team to have a detailed understanding data models and interfaces of service providers and service consumers.
- Do not implement an ESB until you need one, and only implement one once you have a SOA strategic vision and a set of adoption project plans. An ESB is a means to and end and not an end in itself. Delaying and ESB implementation will save resources until such time they are needed an allow time for industry to mature standards and tools for implementing the ESB.

- Adopt and Implement an ESB incrementally to build upon lessons learned.
- Provide a common set of management capabilities for services and endpoints including alerting, statistics, audits, and logging for an ESB.
- Design and implement an ESB to scale beyond the performance requirements of all service providers and consumers deployed within the ESB. XML performance for streaming data and transformation is particularly important. Non-blocking input and output is also required to prevent components from blocking while waiting for other components to respond.
- Design and implement an ESB to s support the overall enterprise security policies for the relevant organizations by incorporating controls for overarching SOA security policies.

Guidance

- G1910: Provide for transformation of **XML** messages using **eXtensible Style Language Transformations** (**XSLT**) when implementing an **Enterprise Service Bus** (**ESB**).
- G1912: Support the execution of a formally specified **Business Process Execution Language** (**BPEL**) when implementing an **Enterprise Service Bus** (**ESB**).

Best Practices

- BP1908: Provide bidirectional mediation between transport protocols mandated in the **Defense IT Standards Registry** (**DISR**) when implementing an **Enterprise Service Bus** (**ESB**).
- BP1909: Provide for filtering of **XML** messages using XML Path Language (XPath) when implementing an **Enterprise Service Bus** (**ESB**).
- BP1911: Provide for routing of messages based on message content when implementing an **Enterprise Service Bus** (**ESB**).
- BP1913: Provide for mediation between synchronous and asynchronous messages when implementing an **Enterprise Service Bus** (**ESB**).

P1087: Software Communication Architecture

The **Software Communications Architecture** (**SCA**) establishes an implementation-independent framework with baseline requirements for the development of software for an established hardware platform, such as software defined radios. The SCA is an architectural framework created to maximize portability, interoperability, and configurability of the software while still allowing the flexibility to address domain specific requirements and restrictions. Constraints on software development imposed by the framework are on the interfaces and the structure of the software and not on the implementation of the functions that are performed.

The framework places an emphasis on areas where reusability is affected and allows implementation unique requirements to determine a specific application of the architecture. SCA specifications incorporate accepted industry standards such as a subset of the **Portable Operating System Interface** (**POSIX**) specification and the **Object Management Group** (**OMG**) **CORBA** specification.[R1109] The Joint Program Executive Office for the **Joint Tactical Radio System** (JPEO **JTRS**) maintains a Standards site with SCA releases and **Application Programming Interfaces** (**APIs**).[R1108]

SCA includes a real-time operating system functionality to provide multi-threaded support for all software executing on the system. Software can include SCA applications, devices, and services. The exact functionality supported by the **Operating Environment** is described by the **Application Environment Profile** (**AEP**) which is a subset of the POSIX specification.

The OMG Domain Special Interest Group for Software Radios (SWRADIO DSIG) and Software Defined Radio Forum (SDRF) are working together toward building an international commercial standard based on the SCA.

The purpose of this perspective is to provide guidance and reference material for Programs providing products and services using SCA in order to increase interoperability and net-centricity.

Guidance

- G1713: Use an **Operating Environment** (**OE**) for all **Software Communications Architecture** (**SCA**) applications that includes middleware which adheres to the Minimum **CORBA** Specification version 1.0.
- G1714: Develop **Software Communications Architecture** (**SCA**) applications to use only **Operating Environment** functionality defined by the SCA Application Environment Profile.

Best Practices

- BP1715: Design **SCA** log services according to the OMG Lightweight Log Service Specification.
- BP1716: Develop applications for **SCA**-compliant systems using a higher order programming language.
- BP1880: Justify, document, and obtain a waiver for all radio terminal acquisitions that are not JTRS/SCA compliant.

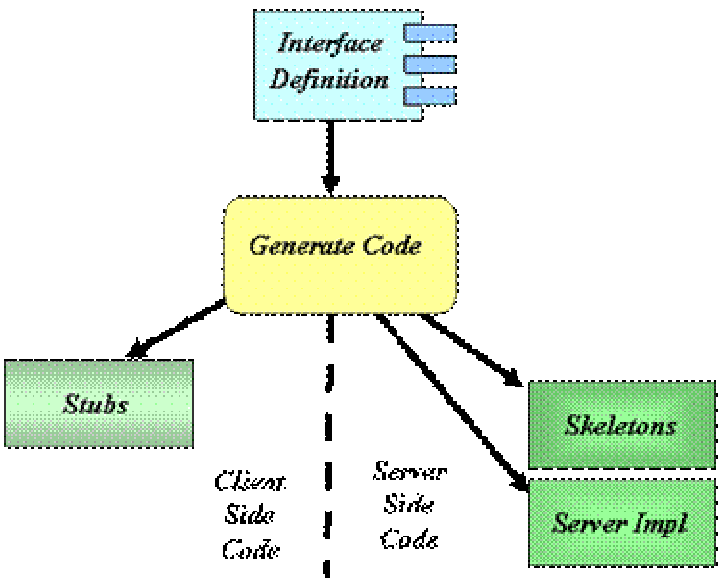
Part 5: Developer Guidance > Middleware > CORBA

P1011: CORBA

CORBA is the acronym for **Common Object Request Broker Architecture**. It is the **Object Management Group** (**OMG**) open, vendor-independent architecture and infrastructure that computer applications use to work together over networks. Using the **Internet InterORB Protocol** (**IIOP**), a CORBA-based program from any vendor, on almost any computer, operating system, programming language, or network, can interoperate with a CORBA-based program from the same or another vendor on almost any other computer, operating system, programming language, or network.

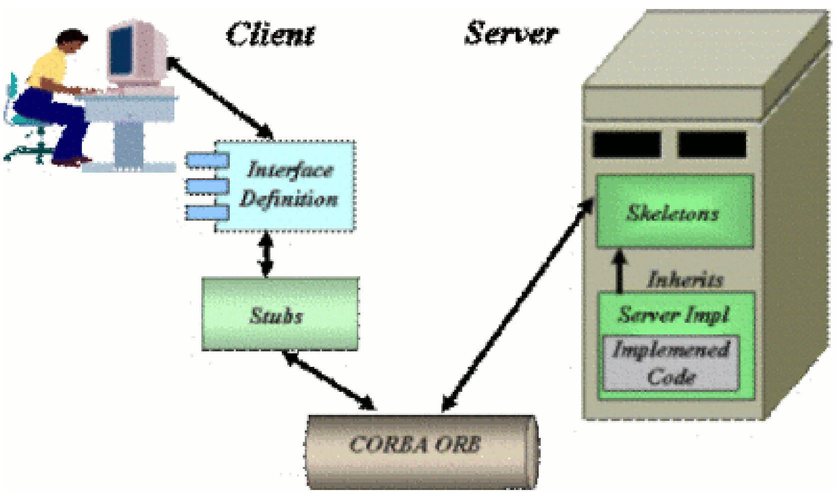
In general, the code that needs to be created to access an object remotely using CORBA can be implemented using well established and well understood design patterns. Consequently, it is not difficult to write but it is tedious and subject to human error during the writing process because much of it is of a cut-and-paste nature. Therefore, most **Object Request Broker** (**ORB**) vendors have developed code generators that can auto-generate the required infrastructure code given the definition of the interface between a **client** and a **server**. The use of these auto-generators is strongly encouraged.

The following diagram illustrates auto-generation of the infrastructure code from an interface defined using the CORBA **Interface Definition Language** (**IDL**).



I1069

This diagram illustrates how the generated code is used within the CORBA infrastructure.



I1071

Key features

Some of the key features of interest in the CORBA specifications follow:

- Internet InterORB Protocol (IIOP)
- Dynamic Invocation Interface (DII)
- Dynamic Skeleton Interface (DSI)
- Interface Repository (IFR)
- Objects by Value (OBV)
- CORBA Component Model (CCM)
- Portable Object Adapter (POA)
- General InterORB Protocol (GIOP)
- Java to Interface Definition Language (IDL) mapping

Guidance

- G1118: Localize **CORBA** vendor-specific source code into separate **modules**.
- G1119: Isolate user-modifiable configuration parameters from the **CORBA** application source code.
- G1121: Do not modify **CORBA** Interface Definition Language (**IDL**) compiler auto-generated stubs and skeletons.
- G1123: Use the Fat Operation Technique in **IDL** operator invocation.
- G1202: Use the **CORBA Portable Object Adapter** (**POA**) instead of the **Basic Object Adapter** (**BOA**).
- G1203: Localize frequently used **CORBA**-specific code in **modules** that multiple applications can use.
- G1204: Create configuration services to provide distributed user control of the appropriate configuration parameters.
- G1205: Use non-source code persistence to store all user-modifiable **CORBA** service configuration parameters.

Best Practices

- BP1231: Use **`CORBA::String_var`** in **IDL** to pass string types in C++.
- BP1232: Do not pass or return a zero or null pointer; instead, pass an empty string.
- BP1233: Do not assign **`CORBA::String_var`** type to **`INOUT`** method parameters.
- BP1234: Assign string values to **`OUT`** , **`INOUT`** , or **`RETURN`** parameters using operations to allocate or duplicate values rather than creating and deleting values.
- BP1235: Assign string values to returned-as-attribute values using operations to allocate or duplicate values rather than creating and deleting values.

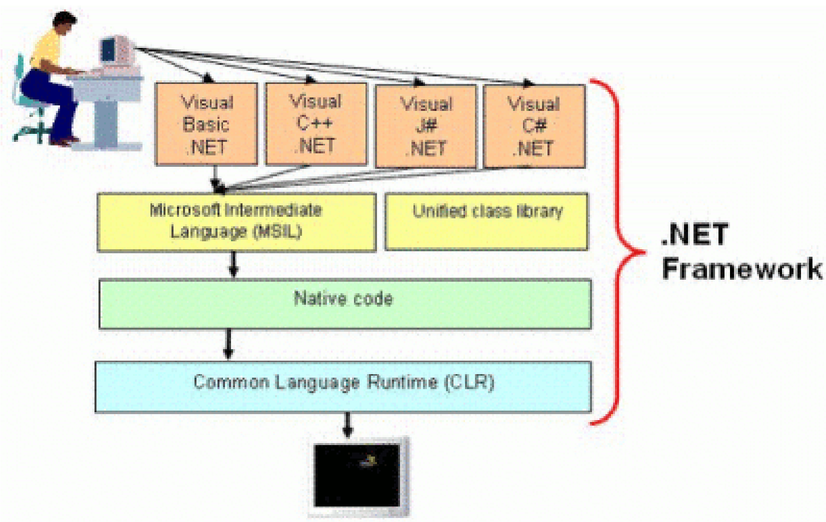
P1086: .NET Framework

To address the confusing maze of computer languages, libraries, tools, and toolkits that were necessary for creating multi-tier applications, Microsoft developed the **.NET Framework** and integrated it into Microsoft Windows as a **component**. It supports building and running multi-tier and **Service-Oriented Architectures** (**SOAs**), including **Web services** and **client** and **server** applications. It simplifies the process of designing, developing, and testing software, allowing individual developers to focus on core, application-specific code.

Microsoft summarizes the .NET Framework as

- A consistent, language-neutral, **object-oriented programming** environment.
- A code-execution environment that minimizes software deployment and versioning conflicts, guarantees safe execution of code, and eliminates the performance problems of scripted or interpreted environments.
- A Common Language Infrastructure (CLI) specification that defines an environment which allows multiple high-level programming languages to be used across different computer platforms without being rewritten for specific architectures.
- A consistent development environment.
- A framework composed of two key parts: an implementation of the CLI called the **Common Language Runtime** (**CLR**) and the **Unified Class Libraries**.

In the Microsoft .NET development environment, a programmer writes software in any one of several Visual .NET languages. These use a single, unified, object-oriented, hierarchical, and extensible set of class libraries to access the system and common services such as **XML** web services, enterprise services, ADO.NET, and XML. Next, the language source code is compiled into an intermediate **Microsoft Intermediate Language** (**MSIL**), which is later translated into platform-specific native code that uses the CLR.



I1064

> ***Note:*** *Microsoft, Hewlett-Packard, and Intel co-sponsored the submission of specifications for the Common Language Infrastructure (CLI) and C# programming language to the international standardization organization Ecma. These specifications are available as Technical Report 84 [R1350] and Technical Report 89 [R1351], respectively. The Mono project is an open source, cross-platform, implementation these specifications that is binary compatible with Microsoft.NET.*

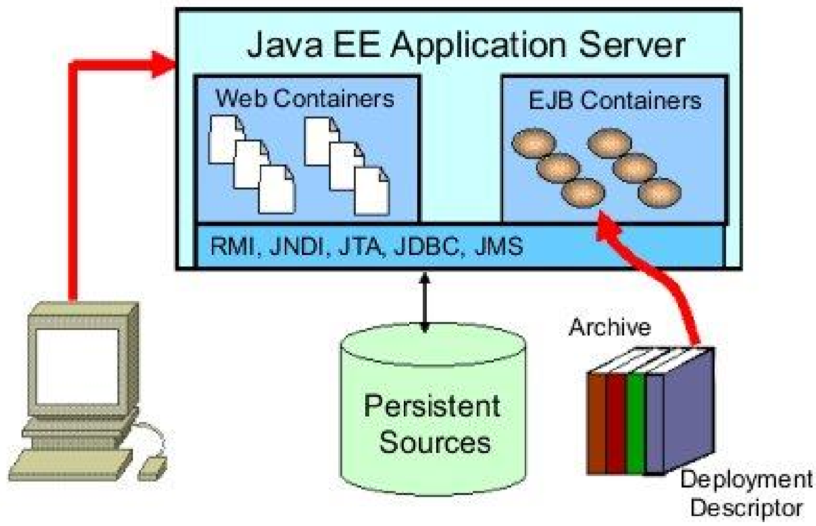
Best Practices

- BP1097: Use the **`System.Text.StringBuilder`** class for repetitive string modifications such as appending, removing, replacing, or inserting characters.
- BP1098: Write all **.NET** code in C#.

- BP1100: Compile all **.NET** code using the .NET **Just-In-Time compiler**.

P1037: Java EE Deployment Descriptors

Java has been extended to handle the complexity of **enterprise** computing through the **Java Enterprise Edition** (**Java EE**, formerly termed **Java 2 Enterprise Edition** or **J2EE**). In the Java EE environment, packaging and **deployment** is done using a Java Archive (JAR) file. A JAR file is a self-contained **module** that contains all of an application's **Java class files**, static files, and **deployment descriptor** files. JAR files are created using a jar utility. There are multiple deployment descriptors that correspond to the type of modules being deployed as indicated in the table below using the Java EE specification.



I1041

The table below shows the Java EE standard deployment descriptor files and the specific applications to which they apply. See http://java.sun.com/dtd/ for details of each XML file.

Component or Application	Scope	Deployment descriptors	Packaging Archives
Web application	Java EE	web.XML	.war
Enterprise bean	Java EE	ejb-jar.XML	.jar
Resource adapter	Java EE	ra.XML	.rar
Enterprise application	Java EE	application.XML	.ear
Client application	Java EE	application-client.XML	

The format for a deployment descriptor is defined in both the **EJB** specification and the **servlet** specification. The Sun standards are defined at the following locations:

Java EE environment applications	http://java.sun.com/products/ejb/docs.html
Non-JavaEE or standard Webapplications	http://java.sun.com/products/servlet/download.html

Note: *Some vendors have extensions to the Java EE deployment descriptors or have specific additional descriptors for their products. Refer to specific vendor documentation for these details.*

Guidance

- G1078: Document the use of non-**Java EE**-defined **deployment descriptors**.
- G1079: Use **deployment descriptors** to isolate configuration data for **Java EE** applications.

P1396: Source Code Migration to Support IPv4 and IPv6

Retrofitting existing **Internet Protocol** (**IP**) version 4 (**IPv4**) software (using existing source code) for IP version independence and developing new IP version-independent software require careful considerations centered on the differences between IPv4 and **IPv6**. Migration also requires a consideration of the reality that IPv4 and IPv6 protocols are going to co-exist on the same platforms and networks for the foreseeable future. In addition, software capable of both IPv4 and IPv6 will have to operate correctly in its deployed environment including IPv4 only environments (due to the possibilities of IPv6 being disabled on the underlying platform or the network's lack of support for IPv6).

In light of this, this perspective and its related guidance aims to aid software developers in developing and migrating source code to support IP versions 4 and 6 by indicating some of the implementation differences between IPv4 and IPv6 **application programming interfaces** (**APIs**) that cause interoperability issues and decrease code portability. This perspective also outlines design impacts arising from supporting one or more version of IP. While some of the content in this perspective only applies to source code containing low-level socket connections, most content contained herein is relevant for all network-enabled source code implemented with any software programming language supporting network socket communications using IP.

This perspective recognizes that many well written sources of IPv4 to IPv6 software migration documentation exist; rather than repeat the information contained in these sources, this perspective provides a summary of common issues and offers additional documentation links for further information and reference such as the following:

- *NESI Part 4: Node Guidance* Internet Protocol [P1139] and IPv4 to IPv6 Transition [P1140] perspectives
- DoD IPv6 Standard Profiles for IPv6 Capable Products Version 4.0 July 2009 (available via the **Joint Interoperability Test Command**)
- Application Aspects of IPv6 Transition (**Internet Engineering Task Force** Request For Comments [RFC] 4038)
- Porting Applications to IPv6 HowTo (Eva M. Castro)
- Porting IPv4 Applications to IPv6 (Sukhdeep Singh Johar)
- Networking IPv6 User Guide for JDK/JRE 5.0 (Sun Microsystems)
- IPv6 Guide for Windows Sockets Applications (MSDN)

Software Transition Scenarios

Flexibility regarding Internet Protocol version is a desirable trait for software as is allows for software to operate in different deployment scenarios. These different deployment scenarios often include heterogeneous environments with varied levels of support for IPv6 in software and hardware as well as various employed transition strategies to include the following:

- dual stack approaches where the operating system, hardware, and network supports simultaneous use of IPv4 and IPv6
- tunneling approaches to allow for transport of IPv6 packets on an IPv4 only network or vice versa
- translation approaches to support an IPv6 only system communicating with an IPv4 only system or vice versa

Therefore, in some cases it is desirable for software to work only with a given protocol version (IPv4 or IPv6) or with both of them. The decision for which IP protocol versions to support is best made as a business decision (based on factors such as schedule, cost, and risk) as well as a technical decision (based on factors such as technical system requirements and the underlying standards used to implement the system).

Four cases (as outlined in RFC 4038) are useful in categorizing the transition of an application:

- Case 1. IPv4-only software deployed in a dual-stack environment. The operating system supports IPv6 hardware, and network, but applications are not yet migrated to support IPv6.
- Case 2. IPv4-only software and IPv6-only software deployed in a dual-stack environment. There are two deployed and maintained versions of an application, one supporting IPv4-only and the other IPv6-only.
- Case 3. Software retrofitted to support both (simultaneously or one at a time) IPv4 and IPv6 in a dual-stack environment.

- Case 4. Software retrofitted to support both (simultaneously or one at a time) IPv4 and IPv6 but only IPv4 is available in the environment; for example, in cases where IPv6 is disabled due to operational and/or developmental considerations.

Case 1 is the current state for many legacy systems that have not undergone software update to support IPv6 as modern hardware and operating systems usually support IPv4 and IPv6. Increasing, networks support both IPv4 and IPv6, and this is the general case for DoD networks. However, this is not the desired state since it has all the shortcoming of IPv4 (such a more limited number of addresses compares to IPv6) and lacks the advantages of IPv6 (such as better support for mobility).

Case 2 is undesirable since it often results in greater costs due to the requirement for two versions of the same software, one to support IPv4 and another to support IPv6. This case also presents challenging configuration management issues with respect to maintaining requirements, patches, deployments, and administration for the two versions of the software.

Case 3 is the desirable state as it provides the greatest flexibility often at less cost with fewer configuration management challenges when compared to case 2. The single software code-base reduces configuration management of the system and, properly implemented, maintains cost efficiency.

Case 4 is a state that systems supporting IPv4 and IPv6 will likely encounter in their deployed environments; systems must support operating in IPv4 only environments or risk loosing network connectivity. IPv6 support may be disabled or not available within an operating system or network due to operational considerations or the current state network migration from IPv4 support to dual stack capable.

Therefore, this perspective takes the approach of providing recommendations that support IPv4 and IPv6 capable source code for deployment in a dual stack environment while supporting fail-over capability to operate in IPv4 only environments. It does this by providing guidance related to support software development in the categories of: general design considerations, address structure considerations, socket functions considerations, and address conversion functions considerations.

General Design Considerations

IPv4 Versus IPv6 Feature Differences

IPv4 and IPv6 do not share the same feature set; some features in IPv4 are not supported in IPv6, while newer IPv6 features are not supported in IPv4. Software developers that use a feature only supported by IPv4 or IPv6, but not both, will have to provide an implementation of the feature for use in the unsupported IP version environments or redesign the software to avoid the use of the feature altogether. However, accommodating the desired feature may require a non-trivial redesign of the software.

One example feature is broadcast support as implemented in IPv4. IPv6 does not provide support for broadcast, but if does provide support for **multicast**. A software developer may choose to use multicast instead of broadcast to support both IPv4 and IPv6 environments. Doing so would generally require less effort than implementing a broadcast-like feature for IPv6; however, some effort will still be requires as the source code must use the proper multicast addresses (IPv4 or IPv6 ones depending on the deployed environment) and socket configuration options as they differ between the two IP versions. Another example feature supported differently by IPv4 and IPv6 is anycast which is new to IPv6 and does not exist in IPv4.

Using Domain Names Rather than IP Addresses

Avoid using IP addresses within source code; use host names which are translated by name resolution functions instead. The use of IP addresses within source code is discouraged for the following reasons:

- IPv4 and IPv6 have different formats for representing IP addresses. Using host names rather than IP addresses simplifies, or eliminates, code that is required to parse destinations.
- IPv6 addresses cannot be read by an IPv4-only device, while an IPv6 device can only read IPv4 addresses if a special address format, which may or may not be employed in allocating addresses, is used.
- IPv6 addresses are much more dynamic that IPv4 addresses and are subject to change at runtime.
- IPv6 promotes the use of multiple IP addresses for each IP network addressable resource. Multiple IP addresses may resolve to the same domain name.

Software should avoid storing or using IP addresses for the aforementioned reasons; store and use domain names rather than IP addresses. IPv4 enabled software designed to store IP addresses should identify and eliminate the storage of IP address in migrating to support IPv6. Replace all pre-stored addresses (including special address such as wildcard, loopback, and broadcast addresses hard-coded within constants) with domain names.

Some systems such as peer to peer systems operating in a tactical environment with highly dynamic mobile networks and intermittent connectivity may need to cache IP addresses for performance reasons or due to the fact that name resolution may not be supported by the underlying network in a sufficient fashion. In such systems consider IP address caches as temporary and not a source of reliable information. In these cases, utilize features such as time to live parameters to purge the IP addresses.

User Interfaces Considerations for IP Addresses

IPv6 address representations are longer than IPv4 addresses and have a different format; software desiring to correctly process such addresses may require changes to support both address types to include changes in memory allocation, parsing routines, and data entry user interfaces. The changes required may be reduced if the software is designed to use host names rather than IP addresses.

Parsing IP Addresses

IPv4 and IPv6 addresses follow different formats. IPv4 addresses are represented as four octets written in decimal notation separated by a period. IPv4 uses a colon to distinguish the port from the IP addresses. In contrast, IPv6 addresses are represented as pairs of octets written in hexadecimal notation separated by a colon. This necessitates parsing routines to handle both types of IP addresses.

IP addresses contained within Uniform Resource Locators (URLs) are a particular problem for parsers, especially for IPv6 addresses. *Format for Literal IPv6 Addresses in URL's* [RFC2732] specifies enclosing literal IPv6 addresses contained within URLs in square brackets ("[ ]").

Another problem with IP address parsers comes when the input is actually a combination of an IP address and port number. With IPv4 these are often coupled with a colon; for example, **`192.0.2.1:80`**. However, this approach would be ambiguous with IPv6, as colons are already used to structure the address. Therefore, the IP address parsers that take the port number separated with a colon should distinguish between IPv4 and IPv6 addresses. A common way to do this is to enclose the address portion in brackets, as is done with URLs.

Some software may also need to specify IPv6 prefixes and lengths. The prefix length should be inserted outside of the address contained within square brackets.

Due to all of these issues with the parsing of IP addresses, the use of address literals is strongly discouraged for general-purpose direct input to the applications. Using host names when possible, rather than IP addresses, simplifies or negates the need to parse IP addresses.

Comparing IP Addresses

Source code that compares IP addresses for equality may require changes to support comparison of IP addresses in software that supports both IPv4 and IPv6. IPv6 and IPv4 address structures differ, and comparing them may cause problems at compile time or runtime. Furthermore, source code modification may be necessary to compare semantically equivalent but representatively different IP addresses, such as an IPv6 address mapped to an IPv4 address, correctly.

Listening For Both IPv4 and IPv6 Connections

Source code needing to listen for both IPv4 and IPv6 connections may do so by listening to a socket bound to a IPv6 wildcard (sometimes called ***anylocal***) address. However such software may be deployed in an environment where support for internal IPv4 mapped IPv6 addresses is disabled (for example due to security concerns). In these environments, the resulting software would not receive both IPv4 and IPv6 connections as expected. A more reliable, although more complicated, option is to create and listen on two separate sockets, one for IPv4-only incoming connections and another for IPv6-only incoming connections.

Handling Multiple IP Addresses

It is common for a given network addressable resource, such as a computer, to have more than one IP address. One reason is unlike IPv4, IPv6 promotes the use of multiple IP addresses per IP enabled device. Another reason is that a given resource may have more than one network interface. A given resource may also have IPv4 addresses as well as IPv6 addresses.

However, communication between two software systems only requires a single source and destination pair. In addition to specifying the communication endpoints, the source and destination IP addresses are a key factor in determining the route of IP packets within a network, especially for IPv6 networks.

Often the underlying operating system determines the source IP address for connections originating from systems with multiple IP addresses based up a best choice configuration. Selecting the destination IP address is usually left to the individual software application. In order to handle communications in a robust way, when connecting to a given hostname which resolves to more than one IP address (which is common for the reasons stated above), software should try connecting to the an address returned by the resolver and should handle failures by trying to connect to another resolved address until a connection is successful or the list is exhausted.

Address Structure Considerations

The longer address length of IPv6 addresses as compared to IPv4 requires a different socket address structure. The socket APIs provide two IP version specific socket address structures, **`sockaddr_in`** and **`sockaddr_in6`** for IPv4 and IPv6 respectively.

Software developers desiring to support both IPv4 and IPv6 can choose to use branching logic in their code and use the appropriate socket address structure; however this may unnecessarily complicate the source code. When writing portable code, it is preferable to eliminate protocol version dependencies from the source code. There is a new data structure, **`sockaddr_storage`**, designed to store all supported protocol-specific address structures and adequately aligned to be cast to the a specific address structure. Hence, software developers should use **`sockaddr_storage`** structure to store IPv4 and IPv6 addresses.

The various socket structures vary is size. Software developers will need to verify that the proper amount of memory is allocated for the appropriate structure in use and verify the memory allocated within any source code that expose or manipulates the size of socket structures or IP addresses.

Macros (for programming languages that support them) sometimes specify socked address structures within source code and are another issue for source code portability. IPv4 APIs use the **`AF_INET`** macro, while IPv6 APIs use the **`AF_INET6`** macro. In order to support both IP versions in a dual stack environment, use the **`AF_UNSPEC`** macro as it provides the flexibility of being able to handle multiple address families to include IPv4 and IPv6.

Socket Functions Considerations

To support IPv6, adaptations to the socket APIs were necessary; the socket function are largely unchanged, but the allowable options to the various function arguments were updated to support IPv6. Software developers need to modify the arguments passed to socket functions to achieve code portability in a dual stack environment. In particular, software developers should use the generic **`sockaddr_storage`** address structure described above for the appropriate socket function arguments used to send or receive packets, as this practice supports operation in a dual stack environment.

Address Conversion Functions Considerations

Software developers should use the new functions **`inet_ntop()`** and **`inet_pton()`** to convert IPv4 and IPv6 addresses from binary to string and string to binary representations. Use the new functions in both IPv4 and IPv6 operations.

Software developers should use **`getaddrinfo()`** and **`getnameinfo()`** to make name and address conversions IP version agnostic. The combination of these two functions allows software to resolve communication protocols supported by the system completely agnostic of IP version.

Best Practices

- BP1834: Develop software to operate in **IPv4**-only environments.
- BP1914: Develop software to operate in dual stack environments.

- BP1924: Develop software to be **IP** version agnostic.
- BP1925: Identify all **IPv4** dependent code in source code.
- BP1926: Eliminate dependencies on a fixed **IP** address.
- BP1927: Use the **`getaddrinfo()`** function when resolving the address of an **IP** host.
- BP1928: Use the **`getnameinfo()`** function when getting the hostname of an **IP** address.
- BP1929: Support the colon (**`:`**) in both **IPv4** and **IPv6** IP addresses.
- BP1930: Identify network addressable resources using host names.
- BP1931: Use generic address structures for both **IPv4** and **IPv6** addresses.
- BP1932: Use an alternative to broadcast addresses in software systems using **IPv6**.

P1448: Logging

Modern software systems are increasingly comprised of modular, often distributed, components. Greater use of abstract **Application Programming Interfaces** (**API**) has lead to further reuse of software components. Software systems today are often composed using open-source, commercial, and custom software. Maintaining consistent logging practices and techniques through a large collection of software components is difficult.

This perspective, to support modularization, improve the usefulness of logging data and the feasibility of aggregated logging data better, provides a set of logging-related Guidance and Best Practice statements as well as supporting contextual information. In particular, this perspective provides guidance for the software developer in order to ensure logging methods within a code-base do not inhibit logging code interoperability or reusability.

> ***Note:*** *The Enterprise Management [P1330] perspective provides an additional discussion of logging and auditing activities from an operational point of view.*

Logging, including the functions of monitoring and auditing logs, is beneficial for identifying security incidents, isolating faults, policy violations, suspicious activity, and proactively identifying operational problems. The following is a partial list of data types to consider for logging in order for logging to be most effective:

- System, infrastructure, application, and component startup and shutdown
- System, infrastructure, application, and component configuration changes
- Changes to persistent data (e.g., additions, edit, and deletions)
- Resource usage statistics such as counter updates, particularly **Service Level Agreement** (**SLA**) thresholds such as high and low thresholds for memory, disk, processor, and bandwidth utilization
- Security exceptions (e.g., Permission Denied, Resource Not Found)
- Authentication and Authorization events (including successful and failed events)
- Errors and exceptional conditions
- Debug and trace information (only if so configured)

In addition to logging the correct sources and types of data, it is important to safeguard security and privacy related information by not exposing sensitive information inappropriately through logging. In cases where there is a requirement to log sensitive security- and privacy-related information, include safeguards (for example, using a security-specific logger to encrypt sensitive logging information) to protect the information from unauthorized, especially unattended exposure. In general, do not log the following types of information without adequate safeguards in place:

- Sensitive security data (e.g., passwords, private keys)
- Privacy-constrained, proprietary, sensitive, or classified data
- Unsanitized user input (in order to prevent security exploits, such as through buffer overflow or injection attacks, and to prevent an attacker from logging deceptive log data)
- Binary data

Logging Abstraction

Modern logging allows for an abstraction of the logging APIs that acts as a stable but customizable **facade** to the rest of the logging infrastructure. This allows for a substitution or reconfiguration of the underlying logging implementation during the deployment of the system. This is important for library or service developers, as the consumer of the library or service can configure the logging implementation to meet the requirements of the overall system.

Most logging facades support the following:

- The ability to swap logging implementations without making changes to source code using the facade or to the facade source code
- The ability to filter logs based on log level through configuration

- Configurable output handlers

Some examples of logging facades follow:

Facade	***Description***	***Supported Languages***
Simple Logging Facade for Java (SLF4J)	SLF4J is a facade various logging frameworks allowing the end user to plug in the desired logging framework at deployment time	Java
Apache Commons Logging	Apache Commons Logging provides a thin adapter allowing configurable bridging to other well known logging systems.	Java
Log4net	Log4net is a port of log4j framework to the .NET runtime. It supports a variety of output targets.	.NET

Logging Implementation and Frameworks

Most logging facades are just a thin wrapper API and require the specifications of an additional logging implementation library to conduct the actual logging, although some logging facades include additional capability to conduct the actual logging along with the facade itself. In either case, there is some overlap in the capabilities provided by the facade and the capabilities provided by good logging implementations. Good logging implementations include the capability to do the following:

- Filter logs based on log level through configuration
- Configure output handlers
- Configure log format and **metadata**
- Configure log persistent storage format to compression, encryption, indexing structure
- Support log rotation and truncation

Implementation Library	***Description***	***Supported Languages***
Log4J	Log4J is an Apache logging API supporting control of log statements output granularity. It is configurable at runtime using external configuration files.	Java (Log4J ports support C, C++, C#, Perl, Python, Ruby, .NET, and Eiffel)
Syslog	Syslog is the de facto logging standard on UNIX-based systems. Syslog is widely implemented and supported by a variety of devices, computing platforms, programming languages, and shell scripting environments.	Various to include C, C++, shell scripting
Pantheios	Pantheios is a platform-independent logging API. It supports filtering of log messages based on severity level and supports configurable output handlers.	C, C++

In addition, most logging implementations support routing logs to an enterprise logging framework such as routing logs to a centralized server. Two standards-based and common logging infrastructures are Syslog and Simple Network Management Protocol (SNMP).

Syslog is a logging implementation as well as a protocol used to transport logging data across a network. Syslog provides a collection of logging related APIs, services, message formats, and protocols for the generation, transport, and storage of logging messages.

SNMP provides standard means for message communication as well as a hierarchical data structure, called a Management Information Base (MIB), that describes status of network accessable devices. SNMP provides a standards-based method to send log messages in the form of a "trap" and is often used in an enterprise logging architecture. SNMP also provides the ability to poll the status of network-accessible resources which is often used to create log messages.

Important Practices to Ensure Usefulness of Logged Data

Logging data is only useful if it is trustable, reconcilable, searchable, and rich enough with data to provide end value while not overwhelming the storage capacity or providing user of the logs with too much information. In other words, logging statements should contain concise data as well as metadata to provide better understanding and context of the data. Logs should be easy to read and parse (for both humans and computers) and not so cryptic or so detailed that important entries are lost or obfuscated.

- Each log entry should contain a timestamp to resolve the sequence of events. The code conducting the logging may provide the timestamp or the logging implementation may provide the timestamp automatically. In some cases, the sequence of events may be captured across multiple logs; thus timestamps should be of sufficient resolution to allow for correlating the log entries. In addition, the time must be synchronized to the same time across all components that conduct logging; see the Network Time Service [P1144] perspective for more information on time synchronization.
- Software developers should use a logging facade that supports multiple logging levels, and each log entry should be assigned to the appropriate log level. Log levels vary between logging framework implementations; log levels such as Error, Warn, Info, Debug, and Trace are typical. Common supported and documented logging levels are preferable to uncommon logging levels that may limit the ability to change the underlying logging implementation. Custom or unusual logging levels may require extra mediation in coalition, joint, or other federated environments.
- It is important to prove contextual information in log entries in order to allow for consumers of the logs to properly analyze the data. Unique identifiers (such as user, transaction, and thread identifiers) are useful information to include when logging.
- Logs should be easy to consume by software as well as people; consequently, most logging implementatins make use of plain text for log entries. When possible, log entries should be a single line in length per log entry. An exception may be when logging exceptions and stack traces. Single line log entries are easier to sort, correlate, and filter with command line tools.
- Developers should rely on the underlying logging implementation to provide metadata about the source of the logging information (for example processing context such as the class name, method name, and line of code). Hard coding this into the logging statement is error prone and becomes a maintenance problem as the code evolves. It is more effective to allow the logging implementation to include this information through configuration of the logging implementation.
- Developers should rely on the underlying logging storage infrastructure implementation to provide for log rotation and not provide custom code to conduct log rotation. Details such as rotation thresholds and schedules are best handled at deployment time and are best implemented through a configuration change in the logging implementation.
- Developers should use an underlying logging implementation that supports multiple concurrent output handlers, which are specified through configuration changes. This allows log entries to be logged to different (sometime simultaneously) locations (such as files, console, database, syslog server, or null output).

Guidance

- G1010: Use a logging facade that allows for specifying the underlying logging framework during software deployment.
- G1340: Log all exceptional conditions.
- G1346: Audit database access.
- G1348: Log database **transactions**.

Best Practices

- BP1715: Design **SCA** log services according to the OMG Lightweight Log Service Specification.

- BP1948: Use a logging facade that supports timestamps.
- BP1949: Write logging entries such that they are a single line in length.
- BP1950: Use a logging facade that supports multiple logging levels.
- BP1951: Use a logging framework that supports log rotation.
- BP1952: Use a logging facade that supports configurable output handlers.

G1001

Use formal standards to define public interfaces.

Rationale:

It is important to use a common language to define the interfaces so producers and consumers can work independently and together.

There are many standards for defining interfaces (**UML**, **WSDL**, and **CORBA**). Use a documented standard that is widely accepted by industry.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Do **UML** documents exist that describe the shared interfaces?

Procedure:

Ask for the design documents to be provided during the review process.

Example:

None.

2) Test:

Are there **WSDL** files that document the interface to Web services?

Procedure:

Look for the existence of `.WSDL` files.

Example:

None.

3) Test:

Are there **IDL** files that document the interfaces to **CORBA** services?

Procedure:

Look for the existence of `.idl` files.

Example:

None.

G1002

Separate public interfaces from implementation.

Rationale:

This guidance encourages clean separation between **interface** and implementation details for all types of application development. This allows components and systems to be **loosely coupled**. The flexibility allows groups of developers to work independently and in parallel to the contract defined by the interface.

Another benefit of hiding implementation details is that it allows the implementation to change without affecting users of the interface. This means the interface can support dynamic and pluggable implementation.

Finally, separating the implementation from the interface allows for version control of the interface separate from the implementation.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Extensibility
NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

C++: Check to make sure interfaces are defined as pure virtual functions.

Procedure:

Make sure **C++** classes are defined in header files. Classes that represent external interfaces should contain only pure virtual functions. Make sure the class does not declare non-constant data members. Also, make sure it does not define default implementation. An interface should provide no default behavior.

Example:

None.

2) Test:

C: Check to make sure functions are declared in a header file using prototypes.

Procedure:

Make sure each library function has a prototype declaration in the header file.

Example:

None.

G1003

Separate shared Application Programming Interfaces (APIs) from internal APIs.

Rationale:

The APIs that are intended to be shared with outside consumers need to remain fairly static in order to facilitate use by the consumers. The consumer and the producer should mutually agree to changes in APIs.

Shared APIs should only have code related to the shared API functionality.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Cross-Security-Domains Exchange
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the API contain extraneous interfaces or code that is not required for the API functionality?

Procedure:

Use coverage tool/Junit to make sure there is no extraneous code.

Example:

None.

G1004

Make public interfaces backward-compatible within the constraints of a published deprecation policy.

Rationale:

The public interface is basically a contract between the producer of the functionality defined in an interface and the consumer of the functionality. This and related guidance statements are intended to ensure that this contract remains intact and that the consumer of the functionality is not broken during the update cycle of the interface.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the public interface (interfaces that are used externally, outside the project's domain) contain versioning information?

Procedure:

Check to make sure the interface/class has versioning information.

Example:

None.

2) Test:

Does the document structure contain a document that indicates the shelf life of deprecated interfaces?

Procedure:

Check for project documents that have information on the life of deprecated interfaces.

Example:

None.

G1008

Isolate the Web service portlet from web hosting infrastructure dependencies by using the Web Services for Remote Portlets (WSRP) Specification protocol.

Rationale:

Insulating platform-specific code (for example code dealing with operating system path conventions) using standard abstractions or custom classes will keep all non-portable code in one place and prevent proliferation of non-portable code throughout the application.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the application contain any platform-specific code that has not been abstracted?

Procedure:

Check code that is non-portable; for instance, the code does not use back slashes (Windows) or forward slashes (UNIX) in literal strings to create a path.

Example:

```
String path = "\tmp";
```

2) Test:

Is platform-specific code isolated into a single class or file?

Procedure:

Search the files for platform-specific code.

Example:

None.

G1010

Use a logging facade that allows for specifying the underlying logging framework during software deployment.

Rationale:

Modern software systems are increasingly comprised of modular, often distributed, components. These components are often based on a number of **Application Programming Interfaces** (**APIs**) as well as open-source, commercial, and custom software. Maintaining consistent logging practices and techniques through a large collection of software components is difficult.

Using a logging facade allows for specifying the underlying logging framework during software deployment which improves code portability and interoperability by allowing for the configuration of the logging implementation to meet the requirements of the overall system.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Enterprise Service Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity
NESI / Part 5: Developer Guidance / Logging

Evaluation Criteria:

1) Test:

Is logging conducted within source code using a logging facade which allows for specifying the underlying logging framework during software deployment?

Procedure:

Examine source code and ensure the code uses a facade to abstract the logging implenentation. Also examine software to ensure that the logging implementation may be specified and configured during deployment.

Example:

None.

G1011

Make components independently deployable.

Rationale:

Independently deployable components do not have any dependencies on other components. This is often unattainable because components are often aggregations of lower-level components. Exceptions to this rule can occur if the relationships between components are one or more of the following:

- well-defined and well thought out
- carefully managed
- externally configurable

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Implement a Component-Based Architecture
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Implement a Component-Based Architecture
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Implement a Component-Based Architecture
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Develop Design Patterns for Data and Services / Implement a Component-Based Architecture
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Evolve Computing Infrastructure / Implement a Component-Based Architecture
NESI / Part 5: Developer Guidance / Implement a Component-Based Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Is the component dependent on other components?

Procedure:

Check for dependencies.

Example:

None.

G1012

Use a set of services to expose component functionality.

Rationale:

By exposing discrete units of functionality as **services**, business and data integrity remain intact. A service receives a request, processes it, and returns the result to the requester as a single operation.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Scalability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Implement a Component-Based Architecture
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Implement a Component-Based Architecture
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Implement a Component-Based Architecture
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Develop Design Patterns for Data and Services / Implement a Component-Based Architecture
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Evolve Computing Infrastructure / Implement a Component-Based Architecture
NESI / Part 5: Developer Guidance / Implement a Component-Based Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are there Web Application Archive (.war) files that contain the component?

Procedure:

Check for the occurrence of **`.war`** files.

Example:

None.

2) Test:

Are there **WSDL** files that define the services?

Procedure:

Check for the occurrence of **`.wsdl`** files.

Example:

None.

G1014

Access databases through open standard interfaces.

Rationale:

The use of non-standard interfaces can cause portability issues. Standards-based database interfaces promote database independence. For example, **Open Database Connectivity** (**ODBC**) is a standard database interface for referencing databases with C/C++ and .NET, while **Java Database Connection** (**JDBC**) is a standard **Application Programming Interface** (**API**) for accessing databases with Java.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are standard interfaces used to access databases?

Procedure:

Check that standards-based interfaces are used to access databases; for example, ODBC for C,C++, or .NET languages, or JDBC for Java.

Example:

None.

G1018

Assign version identifiers to all public interfaces.

Rationale:

Assigning versions is necessary when determining compatibility between the **interface** and its consumer. Versioning public interfaces allows all parties to track the evolution of the interface for backward compatibility. This can help consumers plan for integration and migration. It is important to have the version information in the shared public interface code because it identifies the actual interface to which consumers of the interface will be coding. Another benefit is that it allows tools to generate the documentation automatically so it does not need to be in two places.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the shared public interface code contain versioning information?

Procedure:

Inspect public interfaces or their supporting documentation for version identifiers.

Example:

None.

G1019

Deprecate public interfaces in accordance with a published deprecation policy.

Rationale:

By deprecating instead of removing interfaces, development teams can plan for software migration and continue to run the software with existing (but deprecated) interfaces.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are public interfaces appropriately deprecated?

Procedure:

Check the project documentation for deprecation policy.

Check that interfaces are properly marked and removed according to the deprecation policy.

Example:

None.

G1022

Insulate public interfaces from compile-time dependencies.

Rationale:

Compile-time dependencies bind not only the capabilities of the included library, module or object, but also the limitations and vulnerabilities to the software being compiled. If the compiled software is a module that provides a public interface itself, any other software that uses that public interface also assumes the benefits, constraints and risks of the underlying compile-time dependencies. While this can significantly optimize the performance of a module, it can also make use of the public interface difficult if the constraints include hardware architecture limitations or if the vulnerabilities include predictable memory targets for attacks. Later binding techniques (at link time or better yet, run time) can minimize these exposures and maximize flexibility, robustness, interoperability and maintainability.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Is the packaging or deployment of the public interface self-contained and isolated to only the public interface(s)?

Procedure:

Check to make sure that the jar, library, assembly, and WSDL only contain the agreed-upon public interface (interfaces being shared externally).

Example:

None.

2) Test:

Does the container (jars, libraries, assemblies, WSDL) contain files other than the interface?

Procedure:

Check to make sure the library does not include or rely upon any other files such as resource files, properties files, configuration files, other libraries, XML files, and so on that would force the repackaging of the public interface.

Example:

None.

3) Test:

Are there any outside influences that could affect the packaging of the public interface?

Procedure:

Check the public interface for dependence on resource files, properties files, configuration files, XML files, and other libraries or packages.

Example:

None.

G1027

Internally document all source code developed with Department of Defense (DoD) funding.

Rationale:

Well-documented source code is easier to maintain and enhance over time. It is hard enough to get documentation about software and to keep it up to date. If the documentation is not internal to the source code, the chances that the software is current and up-to-date decreases. In recent years, the trend has been to generate external documentation about the software by processing the source code and comments (e.g., **Javadoc**).

In addition to documenting the functionality of the source code, it is important to capture the configuration control information (e.g., Concurrent Versioning System or CVS, Subversion, and Web-based Distributed Authoring and Versioning or WebDAV).

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Standard Interface Documentation
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Standard Interface Documentation
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Standard Interface Documentation
NESI / Part 5: Developer Guidance / Standard Interface Documentation

Evaluation Criteria:

1) Test:

Do all the source code files have a header that includes configuration information?

Procedure:

Scan each file and make sure the header also includes configuration management information such as author, date created, and a history of modifications and versions.

Example:

None.

2) Test:

Do all the source code files have internal documentation for attributes, methods that a computer process?

Procedure:

Scan the source files and make sure they are internally documented with tags such as Javadoc or XML tags.

Example:

None.

G1030

Use a user interface component library.

Rationale:

User interface component libraries provide a standardized, well-tested look-and-feel without significant development effort. However, care must be taken to ensure that the application code is insulated from dependencies upon a specific UI component library.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Thick Clients
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Thick Clients
NESI / Part 5: Developer Guidance / User Interfaces / Thick Clients
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the application use a user interface component library?

Procedure:

Check for user interface component library code dependencies in the user interface code.

Example:

None.

G1032

Validate all input fields.

Rationale:

Input validation contributes to data integrity, security, and enhances the end-user experience by detecting errors and preventing problems as close as possible to the point of data entry.

Input validation can be simplified by reducing the number of free form text fields and using selection mechanisms such as radio buttons, option boxes, pull down lists, maps, calendars, clocks, slider bars, and other numeric validation entries.

User input data validation should not be the sole mechanism to ensure data integrity. For example, web applications client -side data validation may be done with javascript, but the user (or an intermediary) may modify or remove the javascript without the knowledge of the server-side web application; therefore it is important to validate input data at both the client-side and server-side.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Enterprise Service Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Are all input fields, including non-freeform fields, validated to ensure they can be properly handled across data interfaces: normalized, mediated, and rendered?

Procedure:

Review the code that receives the input fields' data and verify that the inputs are validated against expected interfaces' data models.

Example:

Sample validation techniques:

- validating input data against a white-list of approved values
- validating input data against a black-list of non-allowed values
- validating input data against a regular expression for proper format
- validating input data to not allow inappropriate execution of commands such as used in SQL-Injections attacks

Sample validation tools:

- IBM WebSphere Voice Toolkit VoiceXML validator tool
- Cisco Systems Audium VoiceXML validation for J2EE

G1043

Separate formatting from data through the use of style sheets instead of hard coded HTML attributes.

Rationale:

Formatting information will be located in one location instead of scattered throughout each individual Web page of a Web site. This makes a Web site more maintainable.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are any formatting attributes used in any of the HTML tags?

Procedure:

Search all Web pages and make sure there are no formatting attributes such as align, color, font, or size in any tags.

Example:

None.

G1044

Comply with Federal accessibility standards contained in Section 508 of the Rehabilitation Act of 1973 (as amended) when developing software user interfaces.

Rationale:

Applicable software must comply with Federal standards to enable better application use for those with disabilities.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Designing User Interfaces for Accessibility
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Designing User Interfaces for Accessibility
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Designing User Interfaces for Accessibility
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Designing User Interfaces for Accessibility
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Do all Web documents (e.g. HTML, JSP, ASP, and CSS) follow the Disability Act guidelines?

Procedure:

Check to make sure all Web documents follow the guidelines.

Use available validation tools to validate Section 508 accessibility and WAI accessibility. Go to http://www.contentquality.com/Default.asp to validate the page.

Example:

None.

G1045

Separate XML data presentation metadata from data values.

Rationale:

XML documents should be free of any presentation information and should only contain data. Separating presentation data from content (for example by representing presentation through the use of using **Cascading Style Sheets** and/or XSL transforms) allows multiple presentations for the same content data.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / XML Rendering
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / XML Rendering
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / XML Rendering
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Is presentation information in XML documents?

Procedure:

Does the XML document contain only data?

If the XML document is not an document, does it contain presentation information?

Example:

None.

G1050

In Active Server Pages (Classic ASP), isolate the presentation tier from the middle tier using Component Object Model (COM) objects.

Rationale:

Using Component Object Model (COM) to separate logic code from presentation code in Classic ASP aids maintenance of both the presentation code and the logic code. It improves code readability and allows for separation of duties between those developing middle tier code and those developing presentation tier code. Separation of duties creates a formal interface, which provides input validation (if done right). Adding more sophisticated security controls creates a hardened boundary that further mitigates potential vulnerabilities. Examples include secured user environments and prevention of compromising interactions with unauthorized information or service provider sites masquerading as rendering instructions (i.e., cross-site scripting or XSS attacks).

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Active Server Pages (Classic ASP)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Active Server Pages (Classic ASP)
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Active Server Pages (Classic ASP)

Evaluation Criteria:

1) Test:

Is all the middle tier code isolated from the presentation tier in Classic ASP via COM?

Procedure:

Verify that Classic ASP files do not contain middle-tier code. Instead, this code should be in COM objects referenced from the Classic ASP.

Example:

None.

G1052

Use the code-behind feature in ASP.NET to separate presentation code from the business logic.

Rationale:

Separating presentation code from business logic allows the developers and content designers to work independently. It also makes the code more maintainable because changes in the design elements or business elements do not affect each other.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Active Server Pages for .NET (ASP.NET)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Active Server Pages for .NET (ASP.NET)
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Active Server Pages for .NET (ASP.NET)

Evaluation Criteria:

1) Test:

Is there code in ASP pages?

Procedure:

Check to make sure that ASP files have the code-behind attribute in the first line instead of embedded C# code in the ASP.

Example:

None.

G1053

Do not embed HTML code in any code-behind code used by aspx pages.

Rationale:

Intermixing VB or C# or C++ with presentation code (HTML) makes the code unnecessarily difficult to maintain by both the developer and designer. This is similar in concept to Java's not embedding HTML code in **servlets**.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Active Server Pages for .NET (ASP.NET)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Active Server Pages for .NET (ASP.NET)
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Active Server Pages for .NET (ASP.NET)

Evaluation Criteria:

1) Test:

Check for HTML code in code-behind code.

Procedure:

Check the code-behind file (`.aspx.vb` for example) for any HTML tags.

Example:

None.

G1056

Specify a versioning policy for .NET assemblies.

Rationale:

Versioning assemblies and configuring dependent assemblies allow the **Common Language Runtime** (**CLR**) to load the proper assemblies at runtime for an application. This insulates the application from configuration changes.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Active Server Pages for .NET (ASP.NET)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Active Server Pages for .NET (ASP.NET)
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Active Server Pages for .NET (ASP.NET)

Evaluation Criteria:

1) Test:

Does the application assembly have versioning information?

Procedure:

Check the application assembly manifest for versioning information.

Use the .NET configuration tool to check for versioning policy and versioning information.

Example:

NoneAdded period after "None" in EC Example.

G1058

Use the Model, View, Controller (MVC) pattern to decouple presentation code from other tiers.

Rationale:

Separating data-layer code from presentation-layer code provides the ability to base multiple views on the same model. This is especially important in the enterprise model because often, the user interface varies with the device (browser, mobile phone, thick client, etc.).

Isolating different layers allows changes to occur in each layer without impacting other layers. For instance, if the data layer (model) decides to switch databases, the changes are isolated to the data layer and do not affect the view layer or controller layer.

Lastly, because MVC architecture enforces separation between presentation, processing, and data layer, this allows functionality to be loosely coupled and therefore more suited for reuse.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Active Server Pages (Classic ASP)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Active Server Pages (Classic ASP)
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Active Server Pages (Classic ASP)
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / JavaServer Pages (JSP)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / JavaServer Pages (JSP)
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / JavaServer Pages (JSP)
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Active Server Pages for .NET (ASP.NET)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Active Server Pages for .NET (ASP.NET)
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Active Server Pages for .NET (ASP.NET)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the application enforce clear separation between data layer (model), presentation layer (view), and middle/ business layer (controller)?

Procedure:

Ensure that all page renderings use a Model-View-Controller (MVC) pattern using, for example, **JavaServer Pages** (**JSPs**) and **servlets** or ASP.NET pages and Code Behind files.

Example:

None.

G1060

Encapsulate Java code in JavaServer Pages Standard Tag Libraries (JSTL) when using the code in JavaServer Pages (JSP).

Rationale:

Separating code from presentation allows developers and designers to work independently. It makes the code reusable and more maintainable because it is defined in a tag library.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / JavaServer Pages (JSP)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / JavaServer Pages (JSP)
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / JavaServer Pages (JSP)

Evaluation Criteria:

1) Test:

Do the JSP pages use JavaServer Page Standard Tag Libraries (JSTL) to encapsulate Java code?

Procedure:

Verify that Java Code is encapsulated in JavaServer Page Tag Libraries within JSPs.

Example:

None.

G1071

Use vendor-neutral interface connections to the enterprise (e.g., LDAP, JNDI, JMS, databases).

Rationale:

Increase **portability** and maintainability. Many of the newer connection mechanisms are vendor-neutral. Use these instead of isolation design patterns or vendor-specific connection mechanisms.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Is the connection mechanism vendor-neutral?

Procedure:

Examine the source code for vendor-specific imports or includes. Use only standard APIs.

Example:

None.

G1073

Isolate vendor extensions to enterprise service interfaces.

Rationale:

Vendor extensions are convenient but help create "vendor lock" and reduce vendor neutrality and migration. It is best to avoid these extensions altogether. If that is not possible, then isolate them in an **adapter** or a wrapper-like construct.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are vendor extensions to enterprise services used?

Procedure:

Make sure that no vendor-specific code is included or imported except as part of an adapter or wrapper.

Example:

None.

G1078

Document the use of non-Java EE-defined deployment descriptors.

Rationale:

Deployment descriptors that are not defined by the J2EE specification are not portable between **application servers**. For example, BEA WebLogic has a vendor-specific deployment descriptor called **`weblogic-ejb-jar.xml`** and JBoss has a vendor specific deployment descriptor called **`jboss-jar.xml`** .

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Java EE Deployment Descriptors
NESI / Part 5: Developer Guidance / Middleware / Java EE Deployment Descriptors
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Are all the XML files that are not part of the Java EE specification identified in a delivered document?

Procedure:

Search all XML documents in the META-INF and WEB-INF directories and identify any XML files that are not defined by Java EE. These files should be in a README or other delivered file that describes their purpose:

Example:

Web application	**`WEB-INF/web.xml`**
EJB JAR	**`META-INF/ejb-jar.xml`**
J2EE Connector	**`META-INF/ra.xml`**
Client application	**`META-INF/application-client.xml`**
Enterprise application	**`META-INF/application.xml`**

G1079

Use deployment descriptors to isolate configuration data for Java EE applications.

Rationale:

Do not hard-code tailorable data into source files. The standard location for tailorable data for Java EE applications is in deployment descriptors. Developers should not "reinvent the wheel" by creating a non-standard mechanism for retrieving configurable data. Make tailorable data accessible through application contexts provided by the application **container** (**Java EE application server**).

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Java EE Deployment Descriptors
NESI / Part 5: Developer Guidance / Middleware / Java EE Deployment Descriptors
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Is tailorable data configured using deployment descriptors?

Procedure:

Check the deployment descriptor for instances of tailorable data.

Example:

Name-value pairs such as **environment variables** configured using resource-env-ref elements.

JNDI locations configured using resource-ref elements.

G1080

Adhere to the Web Services Interoperability Organization (WS-I) Basic Profile specification for Web service environments.

Rationale:

Most of the **COTS** Web service products have already met this requirement. This is intended to cause a rejection of the non-standard Web server.

The WS-I Basic Profile specification is available from the Web Services Interoperability Organization Web site: WS-I Org Basic Profile.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / Web Services Compliance
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / Web Services Compliance
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / Web Services Compliance
NESI / Part 5: Developer Guidance / Middleware / Web Services / Web Services Compliance
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Is the Web service product WS-I Basic Profile specification compliant?

Procedure:

Identify the Web service product being used, and verify through a literature search that it is WS-I Basic Profile specification compliant.

Example:

None.

G1082

Use the document-literal style for all data transferred using SOAP where the document uses the World Wide Web Consortium (W3C) Document Object Model (DOM).

Rationale:

The document-literal style requires defining the input and output parameters to a Web service as documents that follow the W3C Document Object Model (DOM). The DOM acts as a contract between the producer and the consumer of the Web service that is formal, well-defined, and rigorous. Validating the DOM against an **XML** Schema Definition (**XSD**) can help resolve discrepancies in the interface.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / SOAP
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / SOAP
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / SOAP
NESI / Part 5: Developer Guidance / Middleware / Web Services / SOAP
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Scalability
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / Web Services Compliance
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / Web Services Compliance
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / Web Services Compliance
NESI / Part 5: Developer Guidance / Middleware / Web Services / Web Services Compliance
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the **WSDL** define input, output, or returned parameters as Documents that follow the **W3C** Document Object Model (**DOM**)?

Procedure:

Review all WSDL files used to describe a Web service, and make sure they only pass documents. Document types should be **`xsd:anyType`**.

Example:

None

G1083

Do not pass Web Services-Interoperability Organization (WS-I) Document Object Model (DOM) documents as strings.

Rationale:

Because of the relative simplicity of converting an **XML** document to a string, it is easy to pass an entire document as a string rather than as an XML document. This can cause problems if the document contains tags that are similar to the tags used in the **SOAP**. Passing it as an XML document ensures that the document is treated as a single entity.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / Web Services Compliance
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / Web Services Compliance
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / Web Services Compliance
NESI / Part 5: Developer Guidance / Middleware / Web Services / Web Services Compliance
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the **WSDL** define input, output, or returned parameters as strings?

Procedure:

Review all the WSDL files used to describe a Web service and make sure that they only pass documents, not strings. Document types should be **`xsd:anyType`**.

Example:

None

G1085

Establish a registered namespace in the XML Gallery in the DoD Metadata Registry for all DoD Programs.

Rationale:

A **registered namespace** permits unique **identification** and categorization of a Program which avoids name collisions and conflicts. The DoD Net-Centric Data Strategy requires storing data products in shared spaces to provide access to all authorized users and tagging these data products with **metadata** to enable discovery of data by authorized users. The use of a unique registered namespace provides an absolute identifier to products associated with a particular product and is an **XSD** schema requirement.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / WSDL
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / WSDL
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / WSDL
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / WSDL
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / WSDL
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / WSDL
NESI / Part 5: Developer Guidance / Middleware / Web Services / WSDL
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the Program have an assigned namespace in the **DoD Metadata Registry**?

Procedure:

Check the **DoD Metadata Registry** to determine whether program is associated with **COI**(s).

Example:

None

G1087

Validate all Web Services Definition Language (WSDL) files that describe Web services.

Rationale:

Manually editing a **WSDL** file is error-prone, work-intensive, and hard to maintain. However, if the user wants to do it, there is no way to detect a manually edited file from one that was auto generated. The important thing is not how the WSDL file is generated but rather that the WSDL file is valid. It must be validated with a WSDL validator.

Note: Not all WSDL files that are generated and valid are necessarily interoperable.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / Insulation and Structure
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / Insulation and Structure
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / Insulation and Structure
NESI / Part 5: Developer Guidance / Middleware / Web Services / Insulation and Structure
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / WSDL
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / WSDL
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / WSDL
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / WSDL
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / WSDL
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / WSDL
NESI / Part 5: Developer Guidance / Middleware / Web Services / WSDL
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services
NESI / Part 5: Developer Guidance / Middleware / Web Services
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Can the **WSDL** file be validated?

Procedure:

Download a validation tool and test WSDL files.

Example:

Sample tools:

WS-I Organization:	http://www.ws-i.org/deliverables/workinggroup.aspx?wg=testingtools
Eclipse:	http://dev.eclipse.org/viewcvs/indextech.cgi/wsvt-home/main.html?rev=1.20
XMethods:	http://xmethods.net/ve2/Tools.po
Pocket Soap:	http://pocketsoap.com/wsdl/

G1088

Use isolation design patterns to define system functionality that manipulates Web services.

Rationale:

Insulating **SOAP** Web-service manipulation using standard abstraction patterns such as a **proxy** or **adapter** insulates the software system from changes in the Web service interface and promotes maintainability.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / Insulation and Structure
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / Insulation and Structure
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / Insulation and Structure
NESI / Part 5: Developer Guidance / Middleware / Web Services / Insulation and Structure
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / SOAP
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / SOAP
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / SOAP
NESI / Part 5: Developer Guidance / Middleware / Web Services / SOAP
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Scalability
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services
NESI / Part 5: Developer Guidance / Middleware / Web Services
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are Web service calls isolated in a single adapter or proxy object?

Procedure:

Check to see if all Web service calls are isolated to a single adapter or proxy object.

Example:

None

2) Test:

Are Web service calls inside of the application code?

Procedure:

Check for proliferation of Web service calls inside an application.

Example:

None

3) Test:

Are SOAP-client calls inside the application code?

Procedure:

Check to see if SOAP-client code is proliferated inside the application code?

Example:

None

G1090

Do not hard-code a Web service's endpoint.

Rationale:

An **endpoint** is the **Uniform Resource Locator** (**URL**) or location of the **Web service** on the **Internet**. A major benefit of Web services is the ability to relocate a Web service to another location or dynamically discover and use a Web service using registry facilities. Hard-coding the URL of the Web service can cause maintenance and portability problems. A better solution to hard-coded endpoints is to provide endpoint **metadata** that is configurable at deployment or during runtime of the service.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / Insulation and Structure
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / Insulation and Structure
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / Insulation and Structure
NESI / Part 5: Developer Guidance / Middleware / Web Services / Insulation and Structure
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services
NESI / Part 5: Developer Guidance / Middleware / Web Services
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are there any hard-coded URLs in the client-side code?

Procedure:

Parse the client code looking for hard-coded URLs.

Example:

None.

G1093

Implement exception handlers for SOAP-based Web services.

Rationale:

SOAP exceptions result when there are connectivity problems or violations in the SOAP protocol between the client and the server.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / SOAP
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / SOAP
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / SOAP
NESI / Part 5: Developer Guidance / Middleware / Web Services / SOAP
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Enterprise Service Management
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the Web application client have exception handlers for **`SOAPExceptions`**.

Procedure:

Check to see that the Web application client has an exception block specifically for **`SOAPException`**.

Example:

None

2) Test:

Does the Web application client test the SOAP response for a fault?

Procedure:

Verify the Web application client handles a true value returned from the **`response.generatedFault`**.

Example:

None

G1094

Catch all exceptions for application code exposed as a Web service.

Rationale:

Any exception can reveal system internals and thus compromise security. Also, internal exceptions are not user friendly.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Enterprise Service Management
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions

Evaluation Criteria:

1) Test:

Does each exposed Web method catch all possible exceptions and re-throw a declared application exception?

Procedure:

Verify that each exposed Web method has an exception block that catches all possible exceptions and then re-throws them as a declared application exceptions.

Example:

None

2) Test:

Does each exposed Web method catch all possible runtime exceptions and re-throw a declared application runtime exception?

Procedure:

Verify that each exposed Web method has an exception block that catches all possible exceptions and then re-throws them as a declared application exceptions.

Example:

None

G1095

Use W3C fault codes for all SOAP faults.

Rationale:

Having predefined and accepted fault codes allows consumers to handle SOAP faults appropriately without prior knowledge of custom fault codes.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / SOAP
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / SOAP
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / SOAP
NESI / Part 5: Developer Guidance / Middleware / Web Services / SOAP
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Enterprise Service Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the Web application throw fault codes from the accepted list of fault codes?

Procedure:

Verify that each fault code thrown by the Web application is from the accepted list of SOAP fault codes defined by the W3C.

Example:

None

G1118

Localize CORBA vendor-specific source code into separate modules.

Rationale:

The general guidance is to minimize CORBA vendor-specific source code, while recognizing that vendor-specific features are necessary in certain circumstances. However, isolating vendor-specific code reduces maintenance effort.

Vendor capabilities tend to change more rapidly than CORBA-standard specifications. Experience shows that vendor updates frequently require modification to application source code, due to changing vendor interface conventions. These modifications impose vendor-version-specific constraints on the application, thereby complicating maintenance.

Example

Encapsulating CORBA ORB operations

The following examples show how to encapsulate binding operations for a C++ ORB, and naming service operations for a Java ORB.

C++ ORB binder template

The code below shows a sample template for binding to the C++ ORB. IONA's ORBIX was used in this example.

```
/* ====================================================
ServerBinder.h (Template)
this is a generic binder to ORBIX
==================================================== */
#ifndef _BINDER_H_
#define _BINDER_H_
#ifndef IOSTREAM_H
#define IOSTREAM_H
#include <iostream.h>
#endif
#ifndef STDLIB_H
#define STDLIB_H
#include <stdlib.h>
#endif
template <class SERVERNAME, class VARPTR>
class Binder
{ private:
    char* serverName;
  public:
    Binder(char* svName):serverName(svName){};
    ~Binder(){};
    int bind( VARPTR* p)
    { int attempts = 0, success = 0;
      int maxtries = 5, retval = 0;
      while ( ( attempts < maxtries )
             && (!success)
            )
      { ++attempts;
        cout << "Binding to server, attempt "
            << attempts
            << endl;
        try
        { (*p) = SERVERNAME::_bind();
          cout << "Bound to server"
              << endl;
          success = retval = 1;
        } // End try
        catch ( CORBA::SystemException &systemException )
        { cout << "SystemException, ServerBinder::bind"
              << endl
              << systemException;
```

```
          success = 1;
          retval = 0;
        } // End catch SystemException
         catch (...)
         { cout << "unknown Exception, ServerBinder::bind"
               << endl;
           success = 1;
           retval = 0;
         } // End catch all
      } //end while
      return retval;
    } //end bind
} //end Binder
#endif
```

Ada ORB binder template for C++

The code below shows a C++ template for binding to an Ada ORB. ORBexpress was used in this example.

```
/* ====================================================
ada_binder.h (Template)
this is a generic binder to ORBExpress
==================================================== */
#ifndef _ADA_BINDER_H_
#define _ADA_BINDER_H_
#ifndef IOSTREAM_H
#define IOSTREAM_H
#include <iostream.h>
#endif
#ifndef STDLIB_H
#define STDLIB_H
#include <stdlib.h>
#endif
template <class SERVERNAME, class VARPTR >
class Ada_Binder
{ private:
    char* adaIorString;
  public:
     Ada_Binder
       ( char* iorString)
       : adaIorString ( iorString )
     {};
     ~Ada_Binder(){};
     int bindToAda( VARPTR* p)
     { int attempts = 0, success = 0;
       int maxtries = 5, retval = 0;
       while ( ( attempts < maxtries)
              && (!success)
             )
       { ++attempts;
         cout << "Binding to server, attempt "
             << attempts
             << endl;
         try
         { cout <<"adaIorString:"
               << endl
               << adaIorString
               << endl;
               (*p) = SERVERNAME::_bind(adaIorString);
//can't use string_to_object in this version
//it kills the ada IOR
//            CORBA::Object_ptr myptr
              CORBA::Orbix.string_to_object
                ( adaIorString );
//            (*p) = SERVERNAME::_narrow(myptr);
            cout << "Bound to server" << endl;
            success = retval = 1;
         } // End try
         catch (CORBA::SystemException& systemException)
         { cout << "SystemException, "
               << "AdaServerBinder::bind"
               << endl
```

```
               << systemException;
           success = 1;
           retval = 0;
         } // End SystemException
         catch (...)
         { cout << "Unknown Exception, "
               << "AdaServerBinder::bind"
               << endl;
           success = 1;
           retval = 0;
         } // End catch all
      } // end while
      return retval;
    } // end bind
} // end ADA_Binder
#endif
```

Example

Naming service operations for a Java ORB

Java helper class

This example is a helper class, JavaNamingHelper.java, that encapsulates CORBA naming service operations for all services to use. We used Java JDK 1.4 ORB to create this example.

```
import java.util.*;
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import org.omg.CORBA_2_3.ORB.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContext.*;
import org.omg.CosNaming.NamingContextPackage.*;
import CBRNSensors.JSLSCAD.*;
public class JavaNamingHelper
{ static NamingContext nameSvc = null;
  static org.omg.CORBA.Object objref = null;
  static JSLSCADSensor myCBRNSensor = null;
  static org.omg.CORBA.Object myobj = null;
  public JavaNamingHelper()
  {
  }
  private static void showNamingContext
    ( org.omg.CORBA.ORB myorb )
  {
  public static NamingContext getNamingSvc
    ( org.omg.CORBA.ORB lclorb,
      String nameSvcName
    )
  { NamingContext lclNameSvc = null;
    try
    { org.omg.CORBA.Object nameSvcObj
        = lclorb.resolve_initial_references
            ( "NameService" );
       // . . . other business logic removed
       //       for brevity
    } // End try
    catch(org.omg.CORBA.COMM_FAILURE cf)
    { . . . // error code goes here
    } // End cstch
    catch ( org.omg.CORBA.ORBPackage.InvalidName invalidName)
    { . . . // error code goes here
    } // End catch
    catch ( SystemException systemException )
    { . . .// error code goes here
    }
  } // End getNamingSvc
  public static org.omg.CORBA.Object getObjFromNameSvc
    ( org.omg.CORBA.ORB myorb,
      String targetSensorName
    )
  { . . . // business logic goes here
```

```
  } //end getObjFromNameSvc
  public static int setObj2NameSvc
    ( org.omg.CORBA.ORB myorb,
      BasesSensor mySensor,
      String targetSensorName
    )
  {. . . // business logic goes here
  }//end setObj2NameSvc
}; //end class JavaNamingHelper
```

Java server implementation

The code below is a sample Java server implementation that uses the naming service helper class.

```
import java.io.*;
import java.util.*;
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import org.omg.CORBA_2_3.ORB.*;
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContext.*;
import org.omg.CosNaming.NamingContextPackage.*;
class MyServer
{ public static Properties props;
  public static ORB myorb = null;
  public static NamingContext nameSvc = null;
  public static RootSensor mySensor = null;
  public static String propertyFilePath = null;
  public static final String MY_SENSOR_NAME = "MYSENSOR";
  static public void main(String[] args)
  { // handle arguments
    System.out.println(" CORBA Server starting...\n");
    try
    { // Initialize the ORB.
      myorb = ORB.init(args, props);
      //instantiate servant and create ref
      POA rootPOA
        = POAHelper.narrow(myorb.resolve_initial_references
           ( "RootPOA" );
     . . . // rest of initialization code goes here
    } // End try
    catch ( org.omg.CORBA.ORBPackage.InvalidName invalidName )
    { . . . //error code goes here
    } // End invalidName
    // other exception types to catch go here
    catch ( SystemException systemException)
    {  System.err.println ( systemException );
    } // End systemException
    // naming service hookup
    JavaNamingHelper.setObj2NameSvc
      ( myorb,mySensor,
        MY_SENSOR_NAME
      );
    try
    { System.out.println(" Ready to service requests\n");
      myorb.run();
    } // End try
    catch(SystemException systemException)
    { System.err.println ( systemException );
    } // End catch systemException
  } // End static block
} // End MyServer
```

Java client implementation

The code below is a sample client implementation that uses the naming service helper class.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Are any non-CORBA compliant CORBA:: objects declared or defined in the module?

Procedure:

Review the code for a service that can be used to obtain configuration.

Example:

None

2) Test:

Does the module contain vendor names anywhere in code text?

Procedure:

Review the code looking for a service that can be used to obtain configuration.

Example:

None

G1119

Isolate user-modifiable configuration parameters from the CORBA application source code.

Rationale:

Configuration parameters control the behavior of the CORBA **ORB** service environment and client/service processes during startup, execution, and termination. This parameterization allows execution-time control modification without having to rebuild, reinstall, or redeploy.

Configuration defines the state of the client-and-service environment throughout the lifetime of the processes involved. This relates to considerations such as the allocation of threading and resources, **POA** policies, the instantiation of servants and their invocations, failure and security behavior, connection management, quality of service prioritization, and so forth. The point is that CORBA provides an extremely complex but flexible environment for distributed computing interaction. Consequently, the designer requires flexible guidance to handle this option-rich environment.

Configuration processes and their related parameters fall into two categories. The first involves configuration matters, which are defined to be perpetually static by the system architecture. The second involves matters that are intended to be modifiable by users.

The first category, immutable configuration settings, relates to fundamental underlying assumptions that are foundational for the implementation. These are matters for which no user modification is ever intended as it would lead to unspecified behavior. Consider the example of a service implementation that is programmed to be single threaded. In this case, multi-threading controls are irrelevant and multiple instantiation would lead to dangerous confusion. For immutable configuration parameters, localized and well-commented implementation in the application source code is appropriate.

For user-modifiable configuration settings, there are two further by-design divisions. The first involves configuration settings that are intended to be accessible by distributed processes. The second involves host-specific settings which relate to resources locally available, for which remote access is not desired. These are discussed in the related sublevel guidance

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Are configuration parameters isolated from CORBA application code?

Procedure:

Check source code for configurable parameters to verify that such parameters are not hard-coded within the code and are configurable within configuration files.

Example:

None.

G1121

Do not modify CORBA Interface Definition Language (IDL) compiler auto-generated stubs and skeletons.

Rationale:

The purpose of the IDL auto-generated stub and skeleton files is to provide a source code facility/mechanism for the developer in a specific language to use the IDL-described object interface in that specific language. The internal content of these files changes with the application's IDL modification, with IDL compiler-environment configuration settings, and with vendor-product compiler and **ORB** upgrades. By design, these files are not intended to be modified by the application developer. Developer modification of any auto-generated stub or skeleton file will typically lead to very severe maintenance hazards and failed application rebuild results.

The stub files describe the language source-code interface from the client side. Their use involves including the client stub header in the application's call invocation code.

The skeleton files describe the language source code interface from the service implementation side. Their use involves including the skeleton header in the application's operator implementation code. Their use also requires developer modification of a renamed clone of the auto-generated skeleton body file. These techniques are described in every ORB vendor's programming reference manuals.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Is any application code contained in the auto-generated code?

Procedure:

Inspect the auto-generated file creation/modification dates to verify that no tampering occurred after the IDL compilation step in the build process.

Example:

The following examples are all based upon a single CORBA IDL interface.

MyIdlInterface.idl

```
interface MyIdlInterface
{
  readonly attribute string version;
  void stop();
  void start();
  string error();
}; // End MyIdlInterface
```

ORBExpress compiler

The ORBExpress IDL compiler generates these files:

- **`myIdlInterface.h`** - Client-side stub header
- **`myIdlInterface.cxx`** - Client-side stub implementation
- **`MyIdlInterface_s.h`** - Abstract servant header
- **`MyIdlInterface_s.cxx`** - Abstract servant implementation
- **`MyIdlInterface_impl.h`** - Server implementation header
- **`MyIdlInterface_impl.cxx`** - Server implementation implementation

> ***Note:*** *The only files that should be edited are **`MyIdlInterface_impl.h`** and **`MyIdlInterface_impl.cxx`** . The IDL compiler checks for the existence of the implementation (i.e. _impl) files and will not overwrite them.*

MyIdlInterface_impl.cxx

```
// Generated for interface MyIdlInterface
// in myIdlInterface.idl
#include "MyIdlInterface_impl.h"
MyIdlInterface_impl::MyIdlInterface_impl
  ( PortableServer::POA* oe_poa,
    const char* oe_object_id
  ) : POA_MyIdlInterface
        ( oe_object_id,
          oe_poa
        )
{ . . . // TO DO: add implementation code here
} // emd constructor
MyIdlInterface_impl::MyIdlInterface_impl
  ( const MyIdlInterface_impl& obj )
  : POA_MyIdlInterface(obj)
{ . . . // TO DO: add implementation code here
} // End constructor
MyIdlInterface_impl::~MyIdlInterface_impl()
{ . . . // TO DO: add implementation code here
} // End destructor
CORBA::Char* MyIdlInterface_impl::version
  ( CORBA::Environment& _env )
{ return CORBA::string_dup(_version);
} // End version
void MyIdlInterface_impl::stop
   ( CORBA::Environment& _env )
{ . . . // TO DO: add implementation code here
} // End stop
void MyIdlInterface_impl::start
   ( CORBA::Environment& _env )
{ . . . // TO DO: add implementation code here
} // End start
CORBA::Char* MyIdlInterface_impl::error
  ( CORBA::Environment& _env )
{ CORBA::Char* result;
  . . . // TO DO: add implementation code here
  return result;
} // End error
```

Java JDK compiler

The Java JDK IDL compiler generates these files:

- **`MyIdlInterface.java`**
- **`MyIdlInterfaceHelper.java`**
- **`MyIdlInterfaceHolder.java`**
- **`MyIdlInterfaceOperations.java`**
- **`MyIdlInterfacePOA.java`**
- **`_MyIdlInterfaceStub.java`**

MyIdlInterfacePOA.java

```
/**
 * MyIdlInterfacePOA.java .
 * Generated by the IDL-to-Java compiler
 * (portable), version "3.1"
 * from myIdlInterface.idl
 */
public abstract class MyIdlInterfacePOA
  extends org.omg.PortableServer.Servant
  implements MyIdlInterfaceOperations,
             org.omg.CORBA.portable.InvokeHandler
{ . . . // rest of the auto-generated code removed for brevity
} // End MyIdlInterfacePOA
```

MyIdlInterfaceImpl.java

```
package myIdlImpl;
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import org.omg.CORBA_2_3.ORB.*;
import org.omg.PortableServer.*;
public class  MyIdlInterfaceImpl
  extends  MyIdlInterfacePOA
{
  private String strVersion;
  private String errString;
  public String version ()
  { . . . // implementation code goes here
    return strVersion;
  } // End version
  public void stop ()
  { . . . // implementation code goes here
  } // End stop
  public void start ()
  { . . . // implementation code goes here
  } // End start
  public String error ()
  {. . . // implementation code goes here
   return errString;
  } // End error
} // End MyIdlInterfaceImpl
```

G1123

Use the Fat Operation Technique in IDL operator invocation.

Rationale:

This reduces the CORBA messaging overhead. The performance cost of network CORBA messaging is determined by two factors: latency and marshaling rate. Call latency is the minimum cost of sending any message at all. The marshaling rate is determined by the sizes of sending and receiving parameters and of return values.

In the situation of a large number of objects involving objects that hold a small amount of stat, the call latency cost far exceeds the marshalling costs. Taking advantage of this reality, the "Fat Operation Technique" involves constructing structure objects which hold an aggregation of related attributes, and using the resulting structures in operation invocation parameters and returns. This amounts to transferring a larger amount of information with each network transaction.

For more information, see "Advanced CORBA Programming with C++" by Henning and Vinoski, 1999 Addison Wesley, Chapter 22.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Scalability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the IDL contain function calls which have structure objects that are passed as parameters or returned from operators?

Procedure:

Inspect the IDL file and manually check for parameters or returns using objects defined as structures, and verify that they are passed from methods also declared in the IDL.

Example:

None

G1125

Use the Department of Defense Metadata Specification (DDMS) for standardized tags and taxonomies.

Rationale:

These standardized tags or Metacards will be developed, maintained, and placed under configuration as appropriate and will comply with the **DDMS** and **COI** guidance. These include specifications defining the tagging for security classification and dissemination control. See the DoD Discovery Metadata Specification Web site (http://metadata.dod.mil/mdr/irs/DDMS/) for the current **DDMS** standards.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Has the Program documented the profile used for published data assets in accordance with guidance?

Procedure:

Check the DoD Metadata Registry to determine whether the program is associated with **COI**(s).

Example:

None

G1127

Use a UDDI specification that supports publishing discovery services.

Rationale:

UDDI provides a registration for services, and the **OASIS** UDDI 2.0 specification has become a standard method for publishing discovery services.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 5: Developer Guidance / Middleware / Web Services / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are the Web services registered in a **UDDI** registry?

Procedure:

Verify the registration in the UDDI registry.

Example:

None

2) Test:

Is the registry **UDDI** 2.0 or higher?

Procedure:

Determine if the particular UDDI registry is UDDI Version 2.0 or higher.

Example:

None

G1131

Use standards-based Universal Description, Discovery, and Integration (UDDI) application programming interfaces (APIs) for all UDDI inquiries.

Rationale:

There is a standard **API** that uses **SOAP** messages to communicate with the UDDI registry. To increase compatibility and portability, use this API exclusively.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 5: Developer Guidance / Middleware / Web Services / Universal Description, Discovery, and Integration (UDDI)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are all the interfaces to the UDDI registry made using the UDDI standard API?

Procedure:

The standard API for UDDI is SOAP based. Requests and responses are passed using documents. Test the traffic flow between the client and the UDDI registry for messages that are defined in the UDDI specification. Use standard libraries to send and receive the messages (e.g., JUDDI for Java).

Checking for the use of packages like JUDDI does not require the application to be running.

Example:

The following is an example as provided in the UDDI API reference: http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm#_Toc25137712 .

find_binding

The find_binding API call returns a **`bindingDetail`** message that contains zero or more binding Template structures matching the criteria specified in the argument list.
Syntax

Syntax

Arguments

serviceKey	This uuid_key is used to specify a particular instance of a businessService element in the registered data. Only bindings in the specific businessService data identified by the serviceKey passed will be searched.
maxRows	This optional integer value allows the requesting program to limit the number of results returned.
findQualifiers	This optional collection of findQualifier elements can be used to alter the default behavior of search functionality. See the findQualifiers appendix for more information.
tModelBag	This is a list of tModel uuid_key values that represents the technical fingerprint of a bindingTemplate structure contained within the businessService specified by the serviceKey value. Only bindingTemplates that contain all of the tModel keys specified will be returned (logical AND). The order of the keys in the tModel bag is not relevant.

find_binding

Arguments

Returns

This API call returns a **`bindingDetail`** message upon success. In the event that no matches were located for the specified criteria, the **`bindingDetail`** structure returned will be empty (i.e., it contains no bindingTemplate data.) This signifies a zero match result. If no arguments are passed, a zero-match result set will be returned.

In the event of an overly large number of matches (as determined by each Operator Site), or if the number of matches exceeds the value of the **`maxRows`** attribute, the Operator site will truncate the result set. If this occurs, the response message will contain the truncated attribute with the value "true".

Caveats

If any error occurs in processing this API call, a **`dispositionReport`** element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed	This signifies that the uuid_key value passed did not match with any known serviceKey or tModelKey values. The error structure will signify which condition occurred first, and the invalid key will be indicated clearly in text.
E_unsupported	This signifies that one of the findQualifier values passed was invalid. The invalid qualifier will be indicated clearly in text.

G1132

Implement the data tier using commercial off-the-shelf (COTS) relational database management system (RDBMS) products that implement a Structured Query Language (SQL).

Rationale:

COTS RDBMS products are technically mature, and their capabilities are continually expanding (to include capabilities such as row-level locking, stored procedures, triggers, and high-level language interfaces). Moreover, there is a large technical community able to develop and maintain data systems based on these products. It is likely that a COTS RDBMS will provide many of the data tier capabilities a developer requires.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Enterprise Service Management
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Is the proposed COTS RDBMS product a readily available and supportable COTS product that implements a Structured Query Language (SQL)?

Procedure:

Verify that the COTS RDBMS product is widely in use in the DoD environment (e.g., Oracle, SQL Server, or DB2), has a large support community, and is likely to be supported for the lifecycle of the project.

Example:

None

G1141

Base data models on existing data models developed by Communities of Interest (COI).

Rationale:

Using COI-developed **data models**, or portions thereof, supports interoperability among systems through the use of common semantics. The use of common semantics aids categorization of data, improving information discovery and use. COI-developed data models are a useful source of common semantics during new and ongoing data modeling efforts.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Accessible
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Be Responsive to User Needs
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling

NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Are data models based on COI-developed data models?

Procedure:

Determine whether a COI exists for the technical areas accommodated in the system requirements. Verify that data models are based on data models the relevant COIs have developed.

Example:

The Universal Core (UCore) data model, Joint Consultation Command and Control Information Exchange Data Model (JC3IEDM), and the National Information Exchange Model (NIEM) are all data models developed through the use of a COI process.

G1144

Develop two-level database models: one level captures the conceptual or logical aspects, and the other level captures the physical aspects.

Rationale:

There are a number of modeling tools available that support entity-relationship diagram (ERD) development. Developers can use these tools to create conceptual/logical models that are independent of the **DBMS** in which the system is implemented and to develop the physical models that are translated directly into data definition language (DDL), the **SQL** code used to create the database. Using a conceptual/logical model permits implementation or reuse of a complex ERD on multiple **DBMS** products.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling

Evaluation Criteria:

1) Test:

Have separate **conceptual**/logical and **physical** models been developed?

Procedure:

Verify the presence of a conceptual/logicalmodel0 and a physical model.

Example:

None.

G1146

Include information in the data model necessary to generate a data dictionary.

Rationale:

A **data dictionary** is an integral part of every system including databases. A description of each data item and the units in which the contents are measured are essential. **Data modeling** tools provide a mechanism for storing information necessary to produce a data dictionary.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test:

Does the data model include description information?

Procedure:

Examine the physical data model.

Example:

None.

G1147

Use domain analysis to define the constraints on input data validation.

Rationale:

Domain analysis is an integral part of any data system including databases. Domains describe the set or range of values that are acceptable for a specific data item. These include, at a minimum the following:

- Data type
- Precision
- Minimum
- Maximum
- Length

These values are used to validate the data.

In the database, the range checking is done via check constraints on the data item. These **check constraints** are generated from the **physical data model** as part of the DDL.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling

NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input

Evaluation Criteria:

1) Test:

Does the data model include constraints derived from domain analysis?

Procedure:

Examine the physical data model.

Example:

None.

G1148

Normalize data models.

Rationale:

Normalization is a central tenet of relational database theory. It is also part of OOA.

A database should usually be normalized to at least third normal form. Although there are seven normal forms, normalization beyond third normal form is rarely considered in practical database design.

Objects developed in the absence of data normalization are prone to unnecessary complexity required to keep multiply copies of data.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling

NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling

Evaluation Criteria:

1) Test:

Is the database design in third normal form?

Procedure:

Examine the conceptual/logical **data model**.

Example:

None

G1151

Define declarative foreign keys for all relationships between tables to enforce referential integrity.

Rationale:

Foreign Key constraints enforce referential integrity. The principle of referential integrity requires that the foreign key values of a child table are either null or match exactly those of the **primary key** in the parent table.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test:

Have foreign-key constraints been incorporated into the database?

Procedure:

Examine the database to determine whether foreign-key constraints have been included in the database creation scripts and created in the database.

Example:

None

G1153

Separate application, presentation, and data tiers.

Rationale:

Separation into tiers allows for the separate maintenance of each tier as long as the interface between tiers does not change. It also allows for multiple implementations of a layer to meet different requirements. This supports technology refresh and certain requirements changes.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Scalability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the program, project or initiative architecture support clear boundaries between application layers, e.g. data, presentation, and business logic layers.

Procedure:

Examine the program, project or initiative architecture and evaluate the degree to which it supports clear boundaries between applications layers such as data, and presentation layers. Verify that the system design accommodates a multi-tier architecture.

Example:

The use of web services is one means of separating the presentation layer from business logic and data layers.

G1154

Use stored procedures for operations that are focused on the insertion and maintenance of data.

Rationale:

Current software design methodologies and architectures call for the implementation of an n-tiered architecture with business rules in the middle tier and data stored in a separate data tier. When multiple applications access a common database, however, the rules may be best located at the data-tier level. Otherwise, changes in one application would have to be coordinated across all applications.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Trustable
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test:

Are database triggers used?

Procedure:

Check for stored procedures that are triggered on insertion, deletion, and update events.

Example:

```
CREATE TRIGGER PersonCheckAge
AFTER INSERT OR UPDATE OF age
ON Person
FOR EACH ROW
BEGIN
  IF (:new.age < 0) THEN
    RAISE_APPLICATION_ERROR
      ( -20000,
        'no negative age allowed'
      );
  END IF;
END;.
```

G1155

Use triggers to enforce referential or data integrity, not to perform complex business logic.

Rationale:

Triggers are fired on events. Current software design methodologies and architectures call for the implementation of an n-tiered architecture with business rules in the middle tier and data stored in a separate data tier. Implementing business logic in triggers, as well as in the middle tier, violates this concept.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Trustable
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Enterprise Service Management

Evaluation Criteria:

1) Test:

Has business logic been incorporated into database triggers?

Procedure:

Examine the database trigger code to determine whether business logic or calls to stored procedures incorporating business logic have been coded into them.

Example:

None

G1190

Use a build tool.

Rationale:

A build tool allows for the encapsulation of building instructions into machine-readable files or sets of files. The instructions can be successfully and consistently repeated.

Referenced By:

NESI / Part 5: Developer Guidance / Automate the Software Build Process
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the program or project use a build tool?

Procedure:

Identify which build tool the program or project is using.

Example:

None.

G1202

Use the CORBA Portable Object Adapter (POA) instead of the Basic Object Adapter (BOA).

Rationale:

The CORBA Basic Object Adapter (BOA) was the CORBA Version 1 specification for the client-server object capability. The BOA specification was found to be so incomplete that vendor-specific interpretations were required for operable implementation. In CORBA Version 2, the Portable Object Adapter (POA) was significantly more complete and flexible. In the current marketplace, POA implementations are standard and, in quality implementations, are not vendor-specific. Consequently, using POA eliminates one significant area of vendor-specific coding.

BOA	***POA***
• Focuses on CORBA server implementations and not CORBA object implementations • Naming convention issues on server side • Tightly coupled to **ORB** implementation • Non-standardized way to connect to ORB • Four activation models for server processes	• Services for lifecycle management • Abstract layer between ORB and object • Standard, portable interface for communicating with ORB runtime • Two servant incarnation styles

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does any CORBA application code reference the **`CORBA::BOA`** identifier.

Procedure:

Review the code for the use of the **`CORBA::BOA`** identifier.

Example:

BOA Coding Example

Client Side

The code below shows a C++ CORBA client BOA initialization for the ORBIX ORB. Other ORB vendors may have different initialization sequences.

```
int main
```

```
  ( int argc,
    char **argv
  )
{ MyServer_var MyVar;
  CORBA::ORB_ptr myOrbPtr
    = CORBA::ORB_init(argc, argv,"Orbix");
  try
  { // The default is the local host:
    MyVar = MyServer::_bind(":ServerName");
  } // End try
  catch ( CORBA::SystemException &sysEx )
  { cerr << "Unexpected system exception" << endl;
    cerr <<&sysEx;
    exit(1);
  } // End CORBA::SystemException
  catch(...)
  { // an error occurred while trying
    // to bind to the grid object.
    cerr << "Bind to object failed" << endl;
    cerr << "Unexpected exception " << endl;
    exit(1);
  } // End catch ...
} // End main
```

Server Side

Use the code below as a model. This example shows a C++ CORBA server BOA init for the ORBIX ORB. For BOA, other ORBS will have a different initialization sequence.

```
try
{ MyObject::myOrb_
    = CORBA::ORB_init(argc, argv, "Orbix");
  MyObject::myboa_
    = MyObject::myOrb_->BOA_init(argc, argv, "Orbix_BOA");
} // End try
catch ( CORBA::SystemException &sysEx )
{ //some exception handling code
} // End catch
try
{ NoeLoggerCfg::myboa_->impl_is_ready("MyServiceName",
  CORBA::ORB::INFINITE_TIMEOUT);
} // End try
catch ( CORBA::SystemException &sysEx )
{ //exception handling code
}
```

POA Coding Example

Client Side

This example shows a C++ CORBA client POA init for the ORBIX ORB. For BOA, other ORBS will have a different initialization sequence.

```
int main
  ( int argc,
    char **argv
  )
{ CORBA::ORB_var myOrb = CORBA::ORB_init(argc, argv);
  try
  { CORBA::Object_var obj
      = ... // however you get the object reference
    if(CORBA::is_nil (obj))
    { cerr << "Nil object reference" << endl;
      throw 0;
    } // End if
  } // End try
  catch ( CORBA::SystemException &sysEx )
  { cerr << "Unexpected system exception" << endl;
    cerr <<&sysEx;
```

```
    exit(1);
  } // End catch CORBA::SystemException
  catch ( ... )
  { cerr << "Unexpected system exception" << endl;
    exit(1);
  } // End catch ...
  myinterface::myobject_var myvar;
  try
  { myvar = myinterface::myobject::_narrow(obj);
  } // End try
  catch ( CORBA::SystemException &sysEx)
  { cerr << "Unexpected system exception" << endl;
    cerr <<&sysEx;
    exit(1);
  } // End catch CORBA::SystemException
} // End main
```

Server Side

Use the code below as a model. This example shows a C++ CORBA server POA init for the ORBIX ORB. For POA, other ORBS will have a different initialization sequence.

```
int main
  ( int argc,
    char *argv[ ]
  )
{ try
  { // initialize the ORB
    orb_var orb  = CORBA::ORB_init(argc, argv, "Orbix");
    // obtain an object reference for the root POA
    object_var obj
      = orb->resolve_initial_references ("RootPOA");
    POA_var poa = POA::_narrow(obj);
    // incarnate a servant
    My_Servant_Impl servant;
    // Implicitly register the servant with the root POA
    obj =  servant._this ();
    //start the POA listening for requests
    poa -> the_POAManager ()->activate ();
    //run the orb's event loop
    orb->run ();
  } // End try
  catch ( CORBA::SystemException &sysEx )
  { // some exception handling code
  } // End catch
} // End main
```

G1203

Localize frequently used CORBA-specific code in modules that multiple applications can use.

Rationale:

In a family of applications, similar patterns of CORBA **Object Request Broker** (**ORB**) invocation sequences frequently arise. This is common in service object initialization, policy association, discovery, binding, and release handling. Implementing this functionality in a utility library paradigm localizes the code to reduce maintenance and facilitate extensibility, and assures consistency across the family of applications.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Extensibility
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Do the standard object policy association CORBA invocations occur in more than one module?

Procedure:

The presence of "**`CORBA::PolicyList`**" in C++ indicates policy presence.

Example:

None

2) Test:

Do the standard object initialization CORBA invocations occur in more than one module?

Procedure:

The presence of "**`CORBA::ORB_var`**" or "**`CORBA::ORB_init`**" in C++ indicates ORB initialization. The presence of "**`CORBA::Object_var`**" in C++ indicates ORB access.

Example:

None

3) Test:

Do the standard object policy association CORBA invocations occur in more than one module?

Procedure:

The presence of "**`CORBA::PolicyList`**" in C++ indicates policy presence.

Example:

None

4) Test:

Do the standard object discovery CORBA invocations occur in more than one module?

Procedure:

The presence of "**`Resolve_NamingService()`**"in C++ indicates intended access to one of CORBA's discovery capabilities.

Example:

None

5) Test:

Do the standard object binding and release CORBA invocations occur in more than one module?

Procedure:

The presence of "**`::_narrow(obj.in())`**" or "**`CORBA::is_nil(`**" in C++ indicates activity associated with obtaining and validating an object binding to a legitimate reference. The presence of "**`CORBA(release)(`**" in C++ indicates intended release of a CORBA-bound object reference.

Example:

None

G1204

Create configuration services to provide distributed user control of the appropriate configuration parameters.

Rationale:

For user-modifiable configuration settings that are intended to be accessible by distributed processes at runtime, the appropriate mechanism for implementation involves **CORBA** services. The first form is a network service to be invoked as a client by the target system application at initialization. This can support a consistent, network-wide distribution of startup parameters. The second form is a service implemented by the target application which allows communication to the application during execution (after startup). This allows **real-time** configuration changes for matters such as **Portable Object Adapter** (**POA**) instantiation threading policies to address load management.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Decentralized Operations and Management
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Is a service defined in the IDL to obtain the configuration parameters?

Procedure:

Review the code for a service that can be used to obtain configuration.

Example:

The following code is an example of a CORBA server that instantiates a configuration service. The service manages the individual configuration parameters for the servers on the ORB.

Ada Example

```
CORBA.ORB.IIOP_English;
pragma Elaborate_All(CORBA.ORB.IIOP_English);
with CORBA ;
with CORBA.BOA ;
with CORBA.ORB ;
with CORBA.Object ;
with Configuration.Impl ;
with Configuration.Helper ;
with Ada.Exceptions ;
with Ada.Text_IO ;
with my_CORBA ;
with Event_Ada_API ;
procedure Configuration_Server is
    -- required for OrbExpress
    First_Variable : CORBA.ORB.Life_Span ;
    -- declare the object instance
    Configuration_Object : Configuration.Ref ;
    --variables needed for ior writing
```

```
    No_Timeout : constant := 0.0;
    Config_Name : constant String
      := Configuration.Helper.Simple_Name ;
    Config_Host : Corba.String ;
    Config_Port : Corba.String ;
begin -- Configuration_Server
  -- create (and initialize) the object
  -- config file is read and the port needed
  -- is in there
  Configuration_Object
    := Configuration.Impl.Create(Config_Name) ;
  GET_HOSTNAME:
  begin
    Config_Host
      := Configuration.Get_String
         ( Self => Configuration_Object,
           Name => Corba.To_Corba_String
                     ( "Local_Host_Shortname" )
         );
  exception -- GET_HOSTNAME
    when others =>
      Ada.Text_IO.Put_Line
        ( "ERROR: Missing parameter"
         & "<Local_Host_Shortname> "
         & "in the config_parameters.txt file."
        );
  end GET_HOSTNAME;
  GET_CS_PORT:
  begin
    Config_Port
      := Configuration.Get_String
         ( Self => Configuration_Object,
           Name => Corba.To_Corba_String
                     ( "Config_Service_Port" )
         );
  Exception -- GET_CS_PORT
    when others =>
      Ada.Text_IO.Put_Line
        ( "ERROR: Missing parameter "
         & "<Config_Service_Port> "
         & "in the config_parameters.txt file."
        );
  end GET_CS_PORT;
  Ada.Text_IO.Put_Line
    ( "Host => "
        & Corba.To_Standard_String(Config_Host)
        & " Port => "
        & Corba.To_Standard_String(Config_Port)
    );
  --timeout 0 so we can write IOR out
  CORBA.BOA.Impl_Is_Ready
      ( Time_Out             => No_Timeout,
        Server_Instance_Name => Config_Name,
        Listen_On_Endpoints  =>
          "tcp://"
          & Corba.To_Standard_String(Config_Host)
          & ":"
          & Corba.To_Standard_String(Config_Port)
       );
      -- ---------------------------------------------
      -- HERE IS WHERE CODE FOR THE IOR TO BE
      -- USED ON THE C++ ORB
      -- ---------------------------------------------
  -- get the IOR and write it to disk
  my_CORBA.Write_IOR_To_File
    ( Server_Name => Config_Name,
      Server_Ref  =>
        CORBA.Object.Ref(Configuration_Object)
    );
  READY_BLOCK:
  begin
    -- notify subscribers of availability
    -- of configuration parameters via the
    -- event service
    Event_Ada_API.Send
```

```
      ( Channel_Name => "Config_Channel",
        Event        => "Configuration Service Ready."
      );
  Exception - READY_BLOCK
    when others =>
      Ada.Text_IO.Put_line
        ( "Configuration_Server : "
         & Exception sending ready signal."
        );
  end READY_BLOCK;
  Ada.Text_IO.Put_line
    ( "Configuration_Server : "
     & Configuration Service Ready."
    );
  CORBA.BOA.Impl_Is_Ready
    ( Time_Out             => CORBA.Infinite_Timeout,
      Server_Instance_Name => Config_Name
    ) ;
exception -- Configuration_Server
  when X_Other: others =>
    Ada.Text_IO.Put_line
      ( "Configuration_Server : "
       & Ada.Exceptions.Exception_Name(X_Other)
      );
end Configuration_Server ;
```

C++ Example

The following code snippets depict a C++ server that instantiates a version collection service for an About box. It uses the IORs from the servers on the Ada ORB via the IOR files, and invokes those objects to get version information. It uses the utility templates for binding. It exemplifies the approach described in Encapsulate CORBA ORB operations for C++.

> ***Note:*** *This was done on the ORBIX C++ and Ada ORBs.*

```
#include <iostream.h>
#include <rw/cstring.h>
#ifndef _STDIO_H
#include <stdio.h>
#endif
#ifndef _STRING_H
#include <string.h>
#endif
#ifndef _STDLIB_H
#include <stdlib.h>
#endif
#ifndef _ASSERT_H
#include <assert.h>
#endif
// Include files for all the objects desired for
// collecting version information
//Ada configuration service
#ifndef configuration_hh
#include <configuration.hh>
#endif
// include files for other desired services;
// removed for brevity
// other support objects and utilities
#ifndef _CORBA_UTILS__
#include <corba_utils.h>
#endif
#ifndef __LOG_API_H__
#include <log_api.h>
#endif
#ifndef _VERSION_AGENT_GLOBALS_H_
#include "version_agent_globals.h"
#endif
const RWCString  Version_Agent_i::MSG_VERSION_NOT_FOUND_
```

```
  = "Version Info. not found for ";
const CORBA::ULong Version_Agent_i::MAXSERVERS_
  = 12;
Version_Agent_i:: Version_Agent_i(): theVersionInfoPtr_(0)
{ theVersionInfoPtr_
     = new versionInfoType(MAXSERVERS_);
  theVersionInfoPtr_->length(MAXSERVERS_);
} // End constructor
Version_Agent_i:: ~Version_Agent_i()
{ // Do nothing
} // End destructor
/***********************************************************
FUNCTION NAME: createVersions
PURPOSE: helper function that gets the version info
INPUT:
OUTPUT:
***********************************************************/
void Version_Agent_i::createVersions ()
{ char *iorString;
  int bBindOk = 0;
  int versionCnt = 0;
  versionInfoType* rl = theVersionInfoPtr_;
  CORBA::ULong MAXSERVERS Version_Agent_i::MAXSERVERS_;
  // server variables for all the objects desired
  // for collecting version information
  // most declarations removed for brevity
    EventServiceFactory_var es_var;
  // Ada configuration service
    Configuration_var cfg_var;
  // === load the versions of the individual components
  // Code for other services removed for brevity
  // This is an ADA service using the IOR string
  {  //****************** config service ***************
    logMsg
      ( "get config service version",
        Log_Api::DEBUG_1_MSG
      );
    RWCString errMsg
      ( Version_Agent_i::MSG_VERSION_NOT_FOUND_.data()
      );
    errMsg.append ( "Configuration Service" );
    // here we get the IOR from the ADA orb using
    // the helper methods
      iorString = getIorFile("Configuration");
    //template class to hide binding issues to the ADA ORB
    If ( iorString )
    { Ada_Binder < Configuration,
      Configuration_var > bo ( iorString );
      bBindOk = bo.bindToAda(&cfg_var) ;
      // get the version info and load it
      If ( bBindOk
          && !( CORBA::is_nil(cfg_var))
         )
     { try
       { char* str = cfg_var->version();
         if ( str )
         { (*theVersionInfoPtr_)[versionCnt]
               = CORBA::string_dup(str);
          delete str;
         } // End if
         else
         { (*theVersionInfoPtr_)[versionCnt]
               = CORBA::string_dup(errMsg.data());
         } // End else
       } // End try
       catch(...)
       { (*theVersionInfoPtr_)[versionCnt]
             = CORBA::string_dup(errMsg.data());
       } // End catch
       cfg_var->_closeChannel();
     } // End if
     else
     { (*theVersionInfoPtr_)[versionCnt]
           = CORBA::string_dup(errMsg.data());
     } // End else
```

```
     if(iorString)
     { free (iorString);
       iorString = NULL;
     } // End if
   } //endif iorstring
   else
   { (*theVersionInfoPtr_)[versionCnt]
       = CORBA::string_dup(errMsg.data());
   } // End else
   //leaving scope releases the corba object
 } //end cfg_svf
 bBindOk = 0;
 versionCnt++;
 assert(versionCnt <= MAXSERVERS);
} // End createVersions
/***********************************************************
FUNCTION NAME: start
PURPOSE:  handle startup specific stuff
INPUT:
OUTPUT:
***********************************************************/
void Version_Agent_i:: start
  ( CORBA::Environment &IT_env
  ) throw (CORBA::SystemException)
{ //get all the version info
  createVersions();
} // End start
/***********************************************************
FUNCTION NAME: stop
PURPOSE:  handle stop specific stuff
INPUT:
OUTPUT:
***********************************************************/
void Version_Agent_i:: stop
  ( CORBA::Environment &IT_env
  ) throw (CORBA::SystemException)
{ // Release info
  // Let CORBA time out the service
  logMsg ( "stop received" );
  VersionAgentGlobals::myboa->setNoHangup ( 0 );
  VersionAgentGlobals::myboa->deactivate_impl
    ( "Version_Agent" );
} //end version impl
```

G1205

Use non-source code persistence to store all user-modifiable CORBA service configuration parameters.

Rationale:

For user-modifiable configuration settings that are host-specific and that are not intended to be accessible by distributed processes at runtime, the appropriate mechanism for implementation involves local persistent storage. The appropriate form of local storage depends on the local host architecture and may be file- or host-DBMS oriented. It is important that such parameters are not stored in source code that requires build processes for modification.

For **SOA** services, configuration parameters relating to invoked services should not be service-host-specific at the invoking client application.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Are there any user-modifiable configuration parameters hard coded in the non-auto-generated files?

Procedure:

Inspect the code for constant strings or constants that contain configuration parameters.

Example:

None.

G1208

Add new functionality rather than redefining existing interfaces in a manner that brings incompatibility.

Rationale:

By not replacing old methods of objects, library functionality consumers can continue to operate and not be forced to upgrade.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are methods that are being replaced marked with deprecated tags?

Procedure:

Check revision history to make sure that methods are deprecated and not removed unless they have expired. "Expired" means that they have passed the expected shelf life, as defined by the project standards or other standards documentation.

Example:

None

2) Test:

Do new methods being added contain information on methods they are replacing?

Procedure:

Check to make sure newly added methods contain information and rationale on the methods they are replacing.

Example:

None

G1213

Provide an architecture design document.

Rationale:

An architectural design document provides evaluators with a roadmap of the application. This helps evaluators verify that the application follows guidance such as using the Model View Controller model.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design

Evaluation Criteria:

1) Test:

Do the project deliverables for evaluation include a document that contains the architectural design of the application?

Procedure:

See if an architectural design document exists.

Example:

None

G1214

Provide a document with a plan for deprecating obsolete interfaces.

Rationale:

This information allows users to phase out deprecated interfaces. For instance, Sun plans to maintain backward compatibility for the **JDK** for seven years. This means developers can count on deprecated methods not being removed for seven years.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design

Evaluation Criteria:

1) Test:

Do the project deliverables for evaluation include a document that contains a plan for deprecating obsolete interfaces?

Procedure:

See if a document with a plan for deprecating obsolete interfaces exists.

Example:

None.

G1215

Provide a coding standards document.

Rationale:

The standards ensure a consistent code base. A coding standards document defines rules to keep code readable, maintainable, and secure.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Secure Coding Standards
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Secure Coding Standards
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Secure Coding Standards
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Secure Coding Standards
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Secure Coding Standards
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Secure Coding Standards

Evaluation Criteria:

1) Test:

Do the project deliverables for evaluation include a coding standards document?

Procedure:

See if a coding standards document exists.

Example:

None

G1216

Provide a software release plan document.

Rationale:

The release plan document ensures that there is a formal process for releasing the software. It includes a description of how to acquire the software from the software configuration management (SCM) repository and how to build, label, and release it.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design

Evaluation Criteria:

1) Test:

Do the project deliverables for evaluation contain a release plan document?

Procedure:

See if a software release plan exists.

Example:

None

G1217

Develop and use externally configurable components.

Rationale:

To be portable and to accommodate reuse, components must be configurable using external descriptors usually defined in **XML**. Examples of things that might need to be configured include the following:

- A data source for the component to obtain a **Java Database Connection** (**JDBC**)
- The location of a service with which the component must communicate
- The location of implementation classes that the component uses

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Implement a Component-Based Architecture
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Implement a Component-Based Architecture
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Implement a Component-Based Architecture
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Develop Design Patterns for Data and Services / Implement a Component-Based Architecture
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Evolve Computing Infrastructure / Implement a Component-Based Architecture
NESI / Part 5: Developer Guidance / Implement a Component-Based Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are deployment descriptors used?

Procedure:

Check for the existence of deployment descriptors in the appropriate directories. Usually the file is named **`web.xml`**.

Example:

None

G1218

Use a build tool that supports operation in an automated mode.

Rationale:

During testing, human interaction can be a cause of error and unrepeatable results. Operating in automated mode can eliminate these errors.

Referenced By:

NESI / Part 5: Developer Guidance / Automate the Software Build Process
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the tool have a build all target?

Procedure:

Check the build scripts or descriptors of the build tool for the ability to build the entire project, system, or application.

Example:

None

G1219

Use a build tool that checks out files from configuration control.

Rationale:

To make sure all the parts of the build are under configuration control, compare all files with the configuration baseline, and download the appropriate files.

Referenced By:

NESI / Part 5: Developer Guidance / Automate the Software Build Process
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the tool have a checkout target?

Procedure:

Check the build scripts or descriptors of the build tool for the ability to check out the entire project, system, or application.

Example:

None

G1220

Use a build tool that compiles source code and dependencies that have been modified.

Rationale:

To limit the changes made between builds, only compile code that has been modified. If there are no intermediate files, then compile all files.

Referenced By:

NESI / Part 5: Developer Guidance / Automate the Software Build Process
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the tool have a compile target?

Procedure:

Check the build scripts or descriptors of the build tool for the ability to compile the entire project, system, or application.

Example:

None

2) Test:

Do all the intermediate files (e.g., **`.obj`** or **`.class`**) have the same date and time stamps?

Procedure:

Scan the files for date and time stamps.

Example:

None

G1221

Use a build tool that creates libraries or archives after all required compilations are complete.

Rationale:

Libraries should be able to be recreated independently of any executables and should always verify that any intermediate files are not stale.

Referenced By:

NESI / Part 5: Developer Guidance / Automate the Software Build Process
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the tool have a generate library target?

Procedure:

Check the build scripts or descriptors of the build tool for the ability to generate the composing libraries or archives.

Example:

None

G1222

Use a build tool that creates executables.

Rationale:

An executable is dependent on many files, including source files, intermediate files, and libraries or archives. The building of the executable must support a control process that includes configuration management, compiling, and testing.

Referenced By:

NESI / Part 5: Developer Guidance / Automate the Software Build Process
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the tool have an executable target?

Procedure:

Check the build scripts or build tool descriptors for the ability to build the executables for the entire project, system, or application.

Example:

None

G1223

Use a build tool that is capable of running unit tests.

Rationale:

All code should be able to be tested independently of creating intermediate files, libraries, or executables.

Tests should be unit tests as well as system-level tests.

Referenced By:

NESI / Part 5: Developer Guidance / Automate the Software Build Process
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the tool have a test target?

Procedure:

Check the build scripts or descriptors of the build tool for the ability to test the entire project, system, or application.

Example:

None

G1224

Use a build tool that cleans out intermediate files that can be regenerated.

Rationale:

For security reasons, all files that comprise the build need to be under configuration control. Cleaning out all files is essential in ensuring that only approved code is incorporated into the build.

Referenced By:

NESI / Part 5: Developer Guidance / Automate the Software Build Process
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the tool have a clean target?

Procedure:

Check the build scripts or descriptors for the build tool for the ability to remove the entire project, system, or application files.

Example:

None

G1225

Use a build tool that is independent of the Integrated Development Environment.

Rationale:

Some build tools are tightly coupled with an **Integrated Development Environment** (**IDE**) that causes vendor lock-in and license issues when the software is delivered to the Government.

Referenced By:

NESI / Part 5: Developer Guidance / Automate the Software Build Process
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Is the build tool one of the recognized standards, such as ant?

Procedure:

Check for files named **`build.xml`**.

Example:

None

2) Test:

Is the build tool one of the recognized standards, such as **`make`** or **`nmake`**?

Procedure:

Check for files with the name **`makefile`**.

Example:

None

3) Test:

Does the build tool require a license?

Procedure:

Check for files with the name **`makefile`**.

Example:

None

G1237

Do not hard-code the configuration data of a Web service vendor.

Rationale:

Some vendors generate code that passes Web service vendor-specific configuration data during initialization or startup. This reduces the portability of the code and can cause maintenance problems later.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Web Services / Insulation and Structure
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Web Services / Insulation and Structure
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Web Services / Insulation and Structure
NESI / Part 5: Developer Guidance / Middleware / Web Services / Insulation and Structure
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Is there any Web service vendor-specific configuration data in the client code?

Procedure:

Parse the code and look for hard-coded configuration data that might be used to configure the vendor's Web service.

Example:

None

G1239

Use design patterns (e.g., facade, proxy, or adapter) or property files to isolate vendor-specifics of vendor-dependent connections to the enterprise.

Rationale:

This isolation increases maintainability. Guidance G1071 asserts that vendor-neutral connection mechanisms should be used. When vendor-specific connection mechanisms are unavoidable, this guidance will apply.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Is the connection mechanism vendor-dependent?

Procedure:

Examine the source code for vendor-specific imports or includes.

Make sure that all references to the vendor-specific connection mechanisms are isolated to a single class (like a helper) or set of methods that are used as part of an isolation design pattern such as facade, proxy, or adapter.

Also, look for hard-coded vendor-specific connection strings.

Example:

None

G1245

Isolate the Web service portlet from platform dependencies using the Web Services for Remote Portlets (WSRP) Specification protocol.

Rationale:

The **OASIS WSRP** 1.0 Specification accounts for the fact that producers and consumers may be implemented on very different platforms, such as a Java EE-based Web service, a Web service implemented on the Microsoft .Net platform, or a **portlet** published directly by a **portal**.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Decentralized Operations and Management
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Web Portals
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Web Portals
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Web Portals
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the Web service implement the WSRP Registration interface?

Procedure:

Look for the occurrence of the **`getService`**, **`register`**, **`deregister`**, and **`modifyRegistration`** methods as defined in the OASIS WSRP Specification.

Example:

```
public static RegistrationService getService
   ( java.lang.String baseEndpoint
   ) throws java.lang.Exception
public RegistrationContext register
  ( java.lang.String consumerName,
    java.lang.String consumerAgent,
    boolean methodGetSupported,
    java.lang.String[] consumerModes,
    java.lang.String[] consumerWindowStates,
    java.lang.String[] consumerUserScopes,
    java.lang.String[] customUserProfileData,
    Property[] registrationProperties
  ) throws java.lang.Exception
public ReturnAny deregister
  ( java.lang.String registrationHandle,
    byte[] registrationState
  ) throws java.lang.Exception
public RegistrationState modifyRegistration
  ( RegistrationContext registrationContext,
    RegistrationData registrationData
  ) throws java.lang.Exception
```

2) Test:

Does the Web service implement the WSRP Service Description interface?

Procedure:

Look for the occurrence of the **`getService`**, **`register`**, and **`getServiceDescription`** methods as defined in the OASIS WSRP Service Description API Specification.

Example:

```
public static ServiceDescriptionService getService
  ( java.lang.String baseEndpoint
  ) throws java.lang.ExceptionThrows:
jpublic ServiceDescription getServiceDescription
  ( RegistrationContext registrationContext,
    java.lang.String[] desiredLocales
  ) throws java.lang.Exception
```

3) Test:

Does the Web service implement the WSRP Portlet Configuration interface?

Procedure:

Look for the occurrence of the **`getService`**, **`getPortletDescription`**, **`clonePortlet`**, **`destroyPortlets`**, **`setPortletProperties`**, **`getPortletProperties`** and **`getPortletPropertyDescription`** methods as defined in the OASIS WSRP Portlet Configuration API Specification.

Example:

```
public static PortletManagementService getService
  ( java.lang.String baseEndpoint
  ) throws java.lang.Exception
public PortletDescriptionResponse getPortletDescription
  ( RegistrationContext registrationContext,
    PortletContext portletContext,
    UserContext userContext,
    java.lang.String[] desiredLocales
  ) throws java.lang.Exception
public PortletContext clonePortlet
  ( RegistrationContext registrationContext,
    PortletContext portletContext,
    UserContext userContext
  ) throws java.lang.Exception
public DestroyPortletsResponse destroyPortlets
  ( RegistrationContext registrationContext,
    java.lang.String[] portletHandles
  ) throws java.lang.Exception
public PortletContext setPortletProperties
  ( RegistrationContext registrationContext,
    PortletContext portletContext,
    UserContext userContext,
    PropertyList propertyList
  ) throws java.lang.Exception
public PropertyList getPortletProperties
  ( RegistrationContext registrationContext,
    PortletContext portletContext,
    UserContext userContext,
    java.lang.String[] names
  ) throws java.lang.Exception
public PortletPropertyDescriptionResponse getPortletPropertyDescription
  ( RegistrationContext registrationContext,
    PortletContext portletContext,
    UserContext userContext,
    java.lang.String[] desiredLocales
  ) throws java.lang.ExceptionThrows
```

4) Test:

Does the Web service implement the WSRP Markup interface?

Procedure:

Look for the definition of the **`getMarkup`**, **`performBlockingInteraction`**, **`initCookie`** and **`releaseSessions`** methods as defined in the OASIS WSRP Markup API Specification.

Example:

```
public MarkupResponse getMarkup
   ( RegistrationContext registrationContext,
     PortletContext portletContext,
     RuntimeContext runtimeContext,
     UserContext userContext,
     MarkupParams markupParams
   ) throws java.lang.Exception
public void performBlockingInteraction
   ( RegistrationContext registrationContext,
     PortletContext portletContext,
     RuntimeContext runtimeContext,
     UserContext userContext,
     MarkupParams markupParams,
     InteractionParams interactionParams
   ) throws java.lang.Exception
public Extension[] initCookie
   ( RegistrationContext registrationContext
   )  throws java.lang.Exception
public Extension[] releaseSessions
   ( RegistrationContext registrationContext,
     java.lang.String[] sessionIDs
   ) throws java.lang.Exception
```

G1267

Use HTML data entry fields on Web pages.

Rationale:

Macromedia Flash and Java Applets can also support data input but are not HTML standards and tend to decrease the maintainability of a Web site.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Do any Web pages have data entry fields?

Procedure:

Search all Web pages for the "applet" and "embed" tags. Load each page found in the search by loading and visually inspecting to see if Flash or Applets are used for data entry.

Example:

Correct Usage:

Incorrect usage:

Applet	
Flash	

G1268

Label all data entry fields.

Rationale:

A label provides the user with a brief description of the text to be entered. Labels are essential for a user to understand the data entry field.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Are all data entry fields labeled?

Procedure:

Search all Web pages for the word "form" and load each resulting Web page in a browser. Visually inspect each data entry field to make sure it has labels.

Example:

None.

G1270

Include scroll bars for text entry areas if the data buffer is greater than the viewable area.

Rationale:

Scroll bars provide a visual cue to the user that the text extends beyond the viewable area. Scroll bars will appear by default for an HTML text area.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Do any Web pages turn off scroll bars for text areas?

Procedure:

Search all Web pages and style sheets for the phrase "overflow:hidden" or a form thereof. This turns off scroll bars using styles, but only works in certain browsers. Make sure it is not used.

Example:

Correct Usage

Scroll bars should not be hidden.

Incorrect Usage

Inline style:

```
<html>
<body>
<form>
<textarea style="overflow:hidden"></textarea>
</form>
</body>
</html>
```

External style:

```
textarea.scroll {
   overflow:hidden;
}
```

G1271

Provide instructions and HTML examples for all style sheets.

Rationale:

An instruction manual will enable developers to use the style sheet correctly and efficiently.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Extensibility
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are instructions included for each style sheet provided?

Procedure:

Verify that a document is provided that contains instructions and example code for each style provided.

Example:

Correct usage:

```
Cascading style sheet:
.td-items {
   text-align:right;
}
```

Example of usage:

Incorrect usage:
No HTML example explaining style usage.

G1276

Do not modify the contents of the Web browser's status bar.

Rationale:

Using the browser's status bar to display text unrelated to status affects interoperability because a user expects the status bar to provide status and nothing else.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Enterprise Service Management
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Do any of the Web pages modify the browser status bar?

Procedure:

Search every Web page for the word "status" and visually inspect each of the search results to see if the status bar has been modified.

Example:

Correct usage:

```
Web pages contain no references to window.status
```

Incorrect usage:

```
window.status = 'text to display in status bar'
```

G1277

Do not use tickers on a Web site.

Rationale:

Tickers can irritate the user and use unnecessary bandwidth.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces

Evaluation Criteria:

1) Test:

Do any Web pages contain scrolling text?

Procedure:

Most tickers are written using Applets or Flash. Search all Web pages for the "applet" and "embed" tags. Load each page found in the search and visually inspect to make sure no tickers exist.

Example:

Correct usage:

```
No applet or flash references contain tickers.
```

Incorrect usage:

Applet:
```
applet code="myticker.class" width="200" height="200"
```
Flash:
```
embed src="myticker.swf" width="200" height="200"
```

G1278

Use the browser default setting for links.

Rationale:

Browsers underline links by default. Do not rely on "mouse over" to identify links. Using mouse over to designate links can confuse and slow down infrequent users because they are uncertain which links perform which functions.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Do any Web pages or style sheets modify the browser default settings for links?

Procedure:

Search all the Web pages and style sheets for "A:link," "A:visited" and "A:active." Inspect all search results and make sure none of them modify the "A:" items.

Example:

Correct usage:

```
Web pages and style sheets should have no reference to A:link, A:visited or A:active.
```

Incorrect usage:

```
A:link, A:visited, A:active {
   text-decoration:none;
}
```

G1283

Use linked style sheets rather than embedded styles.

Rationale:

Only by referencing an external file will you be able to update the look of an entire Web site with a single change. Also, by pulling style definitions out of the pages, they (Web pages) will be smaller and faster to download.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Scalability
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test:

Does a Web page use the LINK tag to include external style sheets instead of embedding styles?

Procedure:

View the source of the HTML page. The header tag (head) should contain links to external style sheet (.css) files. The header tag should not contain any style tags.

Example:

Correct usage:

External style:

```
<head>
  <link rel=stylesheet href="style.css" type="text/css" media=screen>
  <link rel=stylesheet href="basic.css" type="text/css" media=screen>
</head>
```

Incorrect usage:

Embedded style:

```
<head>
  <style type="text/css">
      td {
      background:#ff0;
      }
  </style>
</head>
```

G1284

Use only one font for HTML body text.

Rationale:

Users may not have a wide variety of fonts available in their browser, so it is best to use a single, common font. The general standard is to make body text sans serif since most people find sans serif fonts easier to read on monitors and **serif** fonts better for printed materials.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the HTML or style sheet refrain from using more than one font?

Procedure:

Search all Web pages and style sheets for the word "font." Make sure only one type of font is used for body text. May need to visually inspect Web pages to see if a defined font style is used within the body.

Example:

Correct usage:

Cascading style sheet:

```
body.main {
   font:sans-serif;
}
```

HTML:

Incorrect usage:

Several font styles are used within a body.

G1285

Use relative font sizes.

Rationale:

Relative font sizes make Web sites more accessible and support meeting the requirements of Section 508 of the Rehabilitation Act of 1973. Relative font sizes allow for a low-vision user to enlarge the size of the text. Relative font sizes also support maintainability by not hard coding fixed **font sizes**.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Are any absolute font sizes utilized?

Procedure:

Search all Web pages and style sheets for the word "font." Inspect the results to make sure no fixed fonts are used (e.g., 12pt).

Example:

Correct Usage

Relative or no font sizes settings are used.
Cascading style sheets:

```
p {
   font-size:200%;
}
p {
   font-size:2em;
}
```

Incorrect Usage

Cascading style sheets:

```
p {
   font-size:12pt;
}
```

HTML (the font attribute should not be used at all within HTML code, only external style sheets):

G1286

Provide text labels for all buttons.

Rationale:

Users need to understand the purpose of all buttons. In some cases an image on the button is not sufficient to convey meaning. Screen scrapers used by the visually impaired work better when text labels are available for buttons.

In cases where icons serve as buttons in order to fit within a small display device (such as a personal digital assistant), providing an option to enable text labels (or providing alternate attributes in the case of Web-based interfaces) supports screen scrapers.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Do all buttons have associated text labels?

Procedure:

Inspect the user interface to verify text labels are available for all buttons.
Text labels may optionally be displayed:
- on or near the button
- as a tooltip when the user hovers over a button
- as part of a help system where a user clicks and identify tool and then clicks a button.

Button label text may not be enabled by default on all applications, especially systems with small resolution screens such as PDAs.

Example:

Correct usage:

```
<form action="mailto:me@abc.com"
method="post">
<input type="submit" name="emailbut"
value="Send feedback" />
</form>
```

Incorrect usage (using images only):

```
<input type="image" src="send.gif" name="
emailbut"/>
```

G1287

Provide feedback when a transaction will require the user to wait.

Rationale:

Users may think that the application has stopped running or is malfunctioning.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Enterprise Service Management
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Does the application provide feedback during long processes?

Procedure:

Run the application and observe any processes that take longer than 10 seconds to complete. Observe if any status indication is provided to alert the user of the status.

Example:

None

G1292

Use text-based Web site navigation.

Rationale:

Text-based navigation works better than image-based navigation because it enables users to understand the link destinations. Users with text-only browsers and browsers with deactivated graphics can see only text-based navigation options.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are there any instances where graphics are used for navigation?

Procedure:

Visually inspect all Web pages and make sure navigation elements are textual.

Example:

None

G1294

Provide a site map on all Web sites.

Rationale:

A site map shows explicit organization of the site. Inexperienced users do not readily form a mental model of the way that information is organized in a Web site, making it hard for them to recover from navigational errors.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces

Evaluation Criteria:

1) Test:

Does the Web site have a site map?

Procedure:

Search all Web pages for anything with the name "sitemap," "site map" and "map." Visually inspect the search results to make sure a site map is included.

Example:

None

G1295

Provide redundant text links for images within an HTML page.

Rationale:

Redundant text links for images within an **HTML** page allow users to navigate the **Web page** even if their browsers do not display images (as in situations where the **Web browser** renders content without images due to bandwidth considerations). Screen scrapers that assist the visually impaired also use redundant text links. Images may occur within Web pages as part of the content or navigation controls to include **image maps**.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are alternative text links provided for all HTML page images used for navigation?

Procedure:

Verify that alternative text links are provided for images used for navigation by inspecting the HTML source code and testing the HTML page in a browser with image rendering turned off.

Example:

None.

G1301

Practice layered security.

Rationale:

An application with layered security provides more protection against attacks. Combining multiple layers of security defenses can provide additional protection when one layer is broken.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Practice Defense in Depth
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Practice Defense in Depth
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Practice Defense in Depth
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Practice Defense in Depth
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Practice Defense in Depth
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Practice Defense in Depth
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Layering and Modularity
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Services / Core Enterprise Services (CES) / Overarching CES Issues / CES Definitions and Status
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Services / Core Enterprise Services (CES) / Overarching CES Issues / CES Definitions and Status
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Services / Core Enterprise Services (CES) / Overarching CES Issues / CES Definitions and Status
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Core Enterprise Services / Core Enterprise Services (CES) / Overarching CES Issues / CES Definitions and Status
NESI / Part 4: Node Guidance / Services / Core Enterprise Services (CES) / Overarching CES Issues / CES Definitions and Status
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Integrity

NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Integrity
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Integrity
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Do internal and external API(s) perform security checks?

Procedure:

Make sure layers of API(s) starting from externally accessible API(s) down through the layers of internally accessible API(s) provide sufficient security checks. For example, does each layer of the API perform data validation? If internal API is calling remote services, is the data sufficiently protected from snoopers (e.g., use of secure sockets)?

Example:

None

2) Test:

Does the application handle security when processing data files?

Procedure:

Embed all application specific resources such as graphics, internal application configuration files such as internationalization properties/resources, XML files as part of a signed application deployment file (.jar, .exe, etc.).

Example:

None

G1302

Validate all inputs.

Rationale:

Do not limit input validation to the presentation tier; rather, all external APIs should validate inputs prior to use. This is just one aspect of defense in depth which can prevent many attacks including SQL Injection, Cross-Site Scripting, Buffer Overflows, and Denial of Service.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Integrity / Data, Application and Service Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Integrity / Data, Application and Service Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Integrity / Data, Application and Service Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Integrity / Data, Application and Service Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Integrity / Data, Application and Service Integrity
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Integrity / Data, Application and Service Integrity
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input

Evaluation Criteria:

1) Test:

Does the application provide proper handling for null input?

Procedure:

Check application handling of null values.

Example:

None

2) Test:

Does the application use prefix or postfix validation (asserts) to verify input parameters?

Procedure:

Check application range validation of externally accessible API(s).

Example:

None

G1304

Unit test all code.

Rationale:

A high percentage of all security violations can be attributed to inadequate or non-existent unit testing. Hackers can take advantage of these.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Quality Assurance to Software Development
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Quality Assurance to Software Development
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Quality Assurance to Software Development
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Quality Assurance to Software Development
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Quality Assurance to Software Development
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Quality Assurance to Software Development

Evaluation Criteria:

1) Test:

Does the project unit test the code base?

Procedure:

Use a coverage tool to determine how much of the project's code have been tested.

Check for use of a unit testing framework (JUnit for example).

Example:

None

G1308

Configure Public Key Enabled applications to use a Federal Information Processing Standard (FIPS) 140-2 certified cryptographic module.

Rationale:

The guidance defines the application types required to support DoD class 3 PKI.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Is the application using an approved **Federal Information Processing Standard** (**FIPS**) 140-2 cryptographic **module**?

Procedure:

Check the cryptographic module to see if it is FIPS 140-2 compliant.

Example:

None

G1309

Make applications handling high value unclassified information in Minimally Protected environments Public Key Enabled to interoperate with DoD High Assurance.

Rationale:

This guidance defines the application types required to support DoD High Assurance (Mission Assurance Category I [MAC I]) certificates.

The definition of MAC I is "systems handling information that is determined to be vital to the operational readiness or mission effectiveness of deployed and contingency forces in terms of both content and timeliness. The consequences of loss of integrity or availability of a MAC I system are unacceptable and could include the immediate and sustained loss of mission effectiveness. MAC I systems require the most stringent protection measures. (DoD Instruction 8580.1, ***Information Assurance (IA) in the Defense Acquisition System***, 9 July 2004. [R1199])

> ***Note:*** *This guidance is derived from DoD Instruction 8520.2,* ***Public Key Infrastructure (PKI) and Public Key (PK) Enabling****, 1 April 2004. [R1206]*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Is the application using a High Assurance key material generated in a **Federal Information Processing Standard** (**FIPS**) 140-2 Level 2 validated hardware cryptographic **module**?

Procedure:

Check cryptographic module to see if it is FIPS 140-2 Level 2 compliant.

Example:

None.

G1310

Protect application cryptographic objects and functions from tampering.

Rationale:

If cryptographic objects such as private keys, key store, and CA trusted certificates are not protected, the system is not secure.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Are cryptographic objects protected?

Procedure:

Check that key stores, private keys, and **trust points** are protected.

Verify a documented procedure for creating and documenting the creation of keys exists.
Verify a documented procedure for obtaining certificates exists.
Verify a documented procedure for backing up cryptographic objects exists.

Example:

Use High Security Level setting in Internet Explorer to ensure password protection is used. See https://infosec.navy.mil/PKI/certs.html for software certificate steps. See https://infosec.navy.mil/PKI/cac.html for CAC.

G1311

Use Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS) when applications communicate with DoD Public Key Infrastructure (PKI) components.

Rationale:

These are the DoD approved protocols and the only supported ones.

> ***Note:*** *This guidance is derived from DoD Instruction 8520.2,* ***Public Key Infrastructure (PKI) and Public Key (PK) Enabling****, 1 April 2004.* [R1206]

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the application use only HTTPS to communicate when using DoD PKI?

Procedure:

Have application access the DoD PKI Global Directory Service (GDS) Directory (dod411.gds.disa.mil/) via HTTPS.

Example:

None

G1312

Make applications capable of being configured for use with DoD PKI.

Rationale:

Applications must be configurable to request and install certificates, add **trust points**, and require client authentication.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.4, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Is there a capability to configure the application for use with DoD PKI?

Procedure:

Check to make sure the application is configurable to accept certificates, load key stores, and add **trust points**; this may involve inspecting user and administrator manuals.

Example:

None

G1313

Provide documentation for application configuration for use with DoD PKI.

Rationale:

Correct configuration is required for ensuring security. Without detailed documentation, personnel with limited knowledge of security or PKI will have little chance of keeping the overall system secure. The Navy Public Key Infrastructure training site, https://infosec.navy.mil/PKI/training.html (DoD PKI Certificate required for access), contains links to several configuration guides.

> ***Note:*** *This guidance is derived from the DoD Instruction 8520.2,* ***Public Key Infrastructure (PKI) and Public Key (PK) Enabling****, 1 April 2004.* *[R1206]*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Public Key Infrastructure (PKI) and PK Enable Applications
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Is there documentation (such as Standard Operating Procedures [SOPs]) on how to configure and setup the application to interoperate within the DoD PKI?

Procedure:

Verify by inspection of the SOPs and by a demonstration that the application performs as documented when the configuration guidance is followed.

Example:

Most application manuals have detailed instructions in enabling PKI (either under the heading "enabling SSL" or "certificates").

G1314

Provide applications the ability to import Public Key Infrastructure (PKI) software certificates.

Rationale:

The whole **Public Key Infrastructure** (**PKI**) system is predicated on the use of public-private key pairs. The ability to import (recover) and export (backup) key pairs is critical to a functional PKI application.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.5, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Is the application able to import a software certificate key for backup/recovery purposes?

Procedure:

Have the application import a software certificate key.

> ***Note:*** *Verify the correctness of the imported file through analysis.*

Example:

Internet Explorer can import/export certificates using Tools > Internet Options. Click on Internet tab and then click on Certificates link. Import/Export options are located here.

UNIX-based Web server keys are exported by making a copy of the keys file and placing it in a safe location.

G1316

Ensure that applications protect private keys.

Rationale:

In order for the PKI system to stay secure, the private key must not be compromised. Protecting the private key helps prevent attackers from decrypting secured data communications.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.5, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the application use and store the private key securely?

Procedure:

Check for the following:

- all copies of the private key destroyed when private key operation is complete; for example, check that the private key does not stay in application memory permanently
- the private key is password protected with a strong password
- the **keystore** is password protected with a strong password

Example:

Attempt to view the contents of the private key using a document viewer program.

G1317

Ensure applications store Certificates for subscribers (the owner of the Public Key contained in the Certificate) when used in the context of signed and/or encrypted email.

Rationale:

This will allow other parties to use the public key to encrypt messages sent to the application.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document. Section (4.5), Version 1.0, July 13, 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Cryptography
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Is the public key available from the Directory Server application?

Procedure:

See if it is possible to extract the public key certificate from the Directory Server application.

Example:

None

G1318

Develop applications such that they provide the capability to manage and store trust points (Certificate Authority Public Key Certificates).

Rationale:

This will ensure the certificate is valid and expedite verification of the certificate.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Is the Certificate Authority public key available from the application?

Procedure:

View the application's trust list to verify DoD PKI Class 3 CA certificates are present.

Example:

For Internet Explorer, view the DoD PKI Class 3 CA certificates by selecting **`Tools`**>**`Internet Options`**. Click on the **`Internet`** tab and then click on the **`Publishers`** button. Click on the **`Trusted Root Certification Authorities`** tab and scroll down to verify that the DoD PKI Class 3 CA certificates are present.

Web server Certificate Authority certificates can usually be viewed by the application's GUI. If a GUI is not offered, reference the application's manual concerning certificate management.

G1319

Ensure applications can recover data encrypted with legacy keys provided by the DoD PKI Key Recovery Manager (KRM).

Rationale:

Applications may have the need to decrypt legacy information that the application originally encrypted.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Is the application able to recover legacy encrypted data?

Procedure:

Acquire the legacy key and demonstrate the ability
to decrypt data that is encoded by that key.

Example:

None

G1320

Use a minimum of 128 bits for symmetric keys.

Rationale:

Strong encryption helps to prevent unauthorized data decryption using modern day resources.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Are symmetric key encryption levels at least 128 bit?

Procedure:

Check the server configuration and verify that the symmetric keys being used are at least 128 bit.

Example:

Verified Web server ciphers under the SSL portion of the configuration pages of the administration server.

For Internet Explorer 5.0 and above, click the **`Help`** menu and then click the **`About Internet Explorer`** option. The About box will list the Cipher Strength.

2) Test:

Is the application using domestic (U.S.) grade ciphers?

Procedure:

Verify that the application supports domestic (U.S.) grade ciphers.

Example:

None.

G1321

Enable applications to be capable of performing Public Key operations necessary to verify signatures on DoD PKI signed objects.

Rationale:

An application must verify the digital signature and check its validity against the current **Certificate Revocation List** (**CRL**) maintained by an on-line repository (e.g., **Online Status Check Responder** or **OSCR**).

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the application verify signed objects?

Procedure:

Check that the application validates signed objects against DoD root certificates.

Check that the signing certificate has not been revoked by checking against Certificate Revocation Lists or using the Online Certificate Status Protocol (OCSP).

Example:

Make a back-up copy of the certificate. For Windows based applications, stop the application and edit the signature of the certificate and save the certificate. Start the application back up. The application should fail to start as the signature check will fail.

For validity checking, confirm a validity check of the certificate was performed by viewing the application's audit log.

G1322

Ensure that applications that interact with the DoD PKI using SSL (i.e., HTTPS) are capable of performing cryptologic operations using the Triple Data Encryption Algorithm (TDEA).

Rationale:

Applications must use cryptographic modules approved under **Federal Information Processing Standard** (**FIPS**) 140-2, Level 1.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Mediate Security Assertions
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the application use TDEA for encrypting and decrypting data?

Procedure:

Inspect the application's configuration file to confirm that TDEA is used for encrypting and decrypting data.

Example:

Most server based applications have cipher related information stored under SSL, certificates, or security. Verify that the application is using TDEA.

G1323

Generate random symmetric encryption keys when using symmetric encryption.

Rationale:

If the application can not generate random keys, then it is vulnerable to attacks if attackers can determine the algorithm for generating the random symmetric encryption keys.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the application generate random symmetric encryption keys?

Procedure:

Verify that the random seed is generated (e.g., by viewing the application's vendor documentation).

Example:

Most server based applications either user MOD_SSL or OPEN_SSL. These two toolkits properly use random seed generators.

Apache based servers may require the administrator to type random keystrokes on the keyboard. This process is generating the random seed.

G1324

Protect symmetric keys for the life of their use.

Rationale:

Symmetric key encryption algorithms are based on trivially related keys for both encryption and decryption. The advantage of symmetric key encryption is that it is much less computationally intensive for encryption and decryption compared to asymmetric algorithms. The disadvantage is that the shared symmetric key must be kept secure during storage and transmission.

To prevent disclosure, new symmetric keys are often generated for each unique **session** and exchanged using another encryption algorithm. Store symmetric keys that are used long term carefully to prevent disclosure.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Are symmetric keys stored in unprotected locations?

Procedure:

Check for hard coded symmetric keys in source code or files with weak permissions.

Example:

Symmetric keys should be generated for each session and destroyed when the session is destroyed, never stored in a file with weak permissions or hard coded in source code.

G1325

Encrypt symmetric keys when not in use.

Rationale:

Symmetric keys enable both sides of the conversation to have knowledge of the key for encryption. It can not be given out freely, which means if it is going to be stored for repeated use, it should be encrypted first before storage.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Cryptography
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the application encrypt symmetric keys when not in use?

Procedure:

Check that the application encrypts symmetric keys during storage.

Example:

None.

G1326

Ensure applications are capable of producing Secure Hash Algorithm (SHA) digests of messages to support verification of DoD PKI signed objects.

Rationale:

Symmetric keys enable both sides of the conversation to have knowledge of the key for encryption. It can not be given out freely, which means if it is going to be stored for repeated use, it should be encrypted first before storage.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the application use SHA digest?

Procedure:

Visually validate that the SHA digest is used for symmetric keys.

Example:

Most application servers allow one to configure the hash to SHA1. Please note that the default for most applications is MD5.

G1327

Enable an application to obtain new Certificates for subscribers.

Rationale:

If the application generates subscriber keys, the application shall demonstrate the ability to generate keys, request new certificates, and obtain new certificates through interaction with the DoD PKI. If the generated keys are for encryption applications, the application shall demonstrate its ability to provide keys to the DoD PKI KRM.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.3.2.2, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Can the application request and obtain new certificates for subscribers?

Procedure:

For application servers, verify that the application can successfully request a certificate via the appropriate certificate request page from a DoD PKI CA.

For application servers, verify that the application can successfully download an issued certificate from a DoD PKI CA.

Example:

Instructions in obtaining a DoD PKI certificate for a user are available at https://infosec.navy.mil/PKI/users.html.

Instructions for obtaining a DoD PKI certificate for web servers including Netscape, Lotus, and Internet Information Services (IIS) is available at https://infosec.navy.mil/PKI/training.html.

G1328

Enable an application to retrieve Certificates for use, including relying party operations.

Rationale:

The ability to retrieve certificates from DoD certificate repositories further ensures the authenticity of the certificate .

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.3.2.3, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Can the application retrieve **Certificates** from a DoD PKI certificate repository?

Procedure:

Verify that the application can communicate with a DoD PKI certificate repository such as GDS.

Example:

This test procedure is only required for applications that must send encrypted e-mail. For this scenario, assume that Outlook is used; instructions for using Outlook 2000 are available at https://infosec.navy.mil/PKI/Outlook_2000_0704.pdf

G1330

Ensure applications are capable of checking the status of Certificates using a Certificate Revocation List (CRL) if not able to use the Online Certificate Status Protocol (OCSP).

Rationale:

Applications must verify the validity of the certificate prior to establishing trust with another entity. **CRL** is the legacy mechanism for validating certificates. Applications should favor **OSCP** for new development.

Applications operating in environments with network connectivity to a **CRL distribution point** should be able to obtain a current CRL. Applications should be able, without user intervention, to obtain a current CRL to check the status of a certificate that contains a CRL distribution point extension. Applications with network connectivity unable to find CRL distribution points automatically should be capable of being configured with a distribution point that the application then uses to obtain CRLs as needed.

Systems on DoD networks must use a local Web cache to obtain the latest DoD PKI issued CRL per Joint Task Force Global Network Operations (JTF GNO) Communications Tasking Order (CTO) 07-015 of 11 December 2007 (specifically Task 11; DoD PKI Certificate required for access). Configuration instructions for known Web cache products in use and alternative CRL caching capabilities are available from the following location: https://www.us.army.mil/suite/page/474113 (Army or Defense On Line [AKO or DKO] site registration and DoD PKI Certificate required for access).

> ***Note:*** *This guidance is derived from DoD Instruction 8520.2,* ***Public Key Infrastructure (PKI) and Public Key (PK) Enabling****, 1 April 2004.* *[R1206]*

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Network Connectivity
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Can the application perform Certificate status checking with a CRL?

Procedure:

Verify that the application can download a CRL successfully.

Example:

Visually inspect the application is configured to use CRLs for validity checking. This can be achieved by looking at the directory in which the application stores the CRLs.

G1331

Ensure applications are able to check the status of a Certificate using the Online Certificate Status Protocol (OCSP).

Rationale:

Applications must verify the validity of the certificate prior to establishing trust with another entity. CRL is the legacy mechanism for validating certificates. Applications should favor **OCSP** for new development.

Applications may use an OSC responder to check the status of a particular certificate when the DoD has an operational responder. Applications shall prepare and transmit the request to the responder using HTTP in accordance with the DoD Class 3 PKI Infrastructure Interface Specification.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.3.2.4.2, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Can the application perform **Certificate** status checking with **OCSP**?

Procedure:

Verify that the application can performing OCSP queries to an **OSC** Responder successfully.

Example:

Visually inspect the application is configured to use OCSP for validity checking. This can be achieved by looking at the configuration file to see that the application is configured to use OCSP. One can also visually look at the application's log file to validate that the application is making OCSP queries.

G1333

Only use a Certificate during the Certificate's validity range, as bounded by the Certificate's "Validity - Not Before" and "Validity - Not After" date fields.

Rationale:

Expired certificates should not be accepted except in cases where legacy data was archived.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Do the date and time of the use of the Certificate fall within the Certificate's validity period?

Procedure:

Visually inspect the certificate's validity dates. The certificate should be valid and not expired.

Example:

Each digital certificate has a lifetime. When viewing a certificate, the certificate will have a valid from date and a valid to date. The current date should fall within this range.

G1335

Make applications capable of being configured to operate only with PKI Certificate Authorities specifically approved by the application's owner/managing entity.

Rationale:

Using approved PKI Certificate Authorities ensures certificate authenticity and ensures that the certificate is chained to the issuer. DoD trust points ensure certificates are chained to the issuer of the certificate and are authentic.

For example, DoD applications are configured to use DoD PKI Certificate Authorities only per the DoD Class 3 PKI - Public Key-Enabled Application Requirements Document Version 1.0, 13 July 2000.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Is the application configured to operate only with approved PKI Certificate Authorities?

Procedure:

Visually inspect that only the DoD PKI certificates are trusted by the application.

Example:

Applications typically allow one to view the trust points via the administrative interface to the application. CA certificates are typically located under Certificate Management, SSL, or Security.

G1338

Ensure that Public Key Enabled applications support multiple organizational units.

Rationale:

DoD requirements dictate that certificates shall support multiple organizational units.

> ***Note:*** *This guidance is derived from DoD Instruction 8520.2, Public Key Infrastructure (PKI) and Public Key (PK) Enabling, 1 April 2004.[R1206]*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Certificate Processing
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Can the application process a **Certificate** that contains multiple organizational units in the Distinguished Name?

Procedure:

Visually inspect the DoD PKI CA certificates stored in the application. You will notice that each certificate contains multiple organizational units (OU=DoD, OU=PKI)

Example:

The majority of certificate request forms do not contain entries for multiple organizational units. In this case, include all of the organizational unit information in the single line. For example, for Navy, please enter the following information next to the Organizational Unit line: Navy, OU=DoD, OU=PKI.

Once the certificate is issued, visually inspect this certificate to verify that the certificate contains these Organizational Unit values.

G1339

Practice defensive programming by checking all method arguments.

Rationale:

Data validation is not limited to Graphical User Interfaces. API(s) and library functions are also susceptible to corruption. The integrity of application can benefit from identifying invalid data as early as possible.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input

Evaluation Criteria:

1) Test:

Does the application perform range validation?

Procedure:

Check for unit tests.

Check thrown exceptions.

Purposely send invalid data to API(s) to test the integrity and handling of invalid data.

Example:

None.

G1340

Log all exceptional conditions.

Rationale:

Logging exceptional conditions can help to identify security problems, trace the source of the exception, and trigger security alerts.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 5: Developer Guidance / Logging

Evaluation Criteria:

1) Test:

Does the application perform logging of exceptional conditions?

Procedure:

Check exception handlers for logging support.

Example:

None.

G1341

Use a security manager support to restrict application access to privileged resources.

Rationale:

Desktop applications by default do not install a security manager. Installing a security manager could prevent unsecured access to resources such as the network and file system. Desktop applications can benefit from using a security manager to ensure that resources are protected.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Cross-Security-Domains Exchange
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does an installed security manager restrict application access to privileged resources?

Procedure:

Check application main method for installation of a security manager.

Example:

None.

G1342

Restrict direct access to class internal variables to functions or methods of the class itself.

Rationale:

One of the primary tenets in Object Oriented Programming is encapsulation. Restricting access to internal variables not only secure the Class/Object against corruption (no data validation), it is also a maintenance issue. Hiding the implementation details allows the flexibility of underlying implementation to change.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Do classes directly expose internal data members?

Procedure:

Make sure all internal class variables are declared private or protected.

Example:

None.

G1343

Declare classes final to stop inheritance and prevent methods from being overridden.

Rationale:

Utility classes and classes that do not intend to be extended (classes used for user authentication) should lock down their implementation. Locking implementation can prevent methods from being overridden. Not locking down implementation can cause corruption of internal class data or allow errant code to run. For example, imagine the possibility of a class that performs credit card processing that can be overridden.

Class implementation can be locked down by declaring the class or methods final.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Java Security
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Are sensitive, security related, and utility classes declared final?

Procedure:

Check classes used in Security related processing (authentication, authorization) final keyword.

Check classes that have sensitive data (social security numbers, medical data, and salary information) for final keyword.

Check Utility classes for final keyword.

Example:

None.

G1344

Encrypt sensitive data stored in configuration or resource files.

Rationale:

Sensitive data used for application configuration files (XML), user profiles, or resource files should be protected from tampering. The sensitive data should be encrypted and or a message **digest** or checksum should be calculated to check for tampering. Application should handle generation, accessing and storing data to these files.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Application Resource Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Application Resource Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Application Resource Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Application Resource Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Application Resource Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Application Resource Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Cryptography
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Is sensitive data in configuration files and user profiles?

Procedure:

Check properties files, XML configuration files or user profiles for sensitive data in the clear.

Check for an application to edit, and creation of the file.

Example:

None.

G1346

Audit database access.

Rationale:

Auditing is critical for data access traceability. If the RDBMS was attacked, auditing is essential not only for figuring out what had occurred but also to recover lost data. Database access auditing provides logs for each access or change to the database by a given user (or an IP address for systems without user authentication).

Often current middle tier technologies (e.g., J2EE, .Net, CORBA, etc.) share database connections and may only have a single database user. Thus the burden is on the middle tier to know the identity of each user and be able to pass this information on the database (e.g., design each table to have data items such as updated by, created by, etc.).

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 5: Developer Guidance / Logging

Evaluation Criteria:

1) Test:

Does the application database include actual user rather than database connection owner?

Procedure:

Check system documentation, database tables, and audit logs to verify that database access audit entries are created for each database access.

Example:

None

G1347

Secure remote connections to a database.

Rationale:

Just because the database is behind the corporate firewall does not mean someone inside the firewall cannot access or listen in on the wire.

Net-centricity implies that a database should be on the network and not constrained to be sitting behind an application server. This means that many unanticipated users may eventually access the database. Thus, database security should not be based on isolation.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Decentralized Operations and Management
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Is data exchanged between the database and client secure?

Procedure:

Check for secure protocol (e.g., SSL) between application and database.

Check for secure data access by IP address.

Check for configuration in the database (user) which limits user from a specified host.

Example:

None.

G1348

Log database transactions.

Rationale:

Transaction logging is generally handled by the database management system and records all changes made to the database, critical for data recovery and traceability.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 5: Developer Guidance / Logging

Evaluation Criteria:

1) Test:

Are database transactions logged?

Procedure:

Commercial database management systems have a feature to log database transactions. Check to determine whether the feature has been turned on in the database management system.

Example:

None.

G1349

Validate all input that will be part of any dynamically generated SQL.

Rationale:

Not validating or filtering parameters used in dynamically generated SQL statements can lead to SQL injection attacks.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input

Evaluation Criteria:

1) Test:

Does the database use filtering or data validation code?

Procedure:

Filter out character like single quote, double quote, slash, back slash, semi colon, extended character like NULL, carry return, new line, etc, in all input strings.

Example:

G1350

Implement a strong password policy for RDBMS.

Rationale:

Clean database installation often contains no passwords for root users. Also, new user accounts often defaults to no password or standard password. Having no passwords allows users access any data. Database users should always be given strong passwords. This implies a non null password, locking unused user accounts and ensuring that system user accounts are not using default passwords

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the database user table include passwords?

Procedure:

Check for null or empty values for passwords in the user table.

Use a commercially available or open source default password analysis tool to ensure that all user accounts do not retain default passwords and to ensure that all passwords are strong.

Example:

None.

G1351

Enhance database security by using multiple user accounts with constraints.

Rationale:

Constrain access to individual tables and functions by creating multiple user accounts for an application and constraining the accounts to specific functions. As a general policy, user accounts should be constrained to the minimal required database access. For example, creation of a read only account should be constrained by granting only select on the tables of interest to the read only user. This aids in password management as well as limiting the potential impact of SQL injection attacks. By granting only insert on a table, for example, and not granting select, the user could in effect create a write only database.

Each application will have different requirements in regards to grants and access to tables. If one application is compromised, it will not affect the other applications.

It also has traceability to determine which application has allowed a security violation.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does each database application user have account constraints in accordance with the user function?

Procedure:

Check each database application user to ensure that the account constraints are in accordance with the user function and do not have unwarranted privileges. For example, check that read only application user accounts have only read access enabled.

Example:

None.

G1352

Use database clustering and redundant array of independent disks (RAID) for high availability of data.

Rationale:

Database clusters combined with RAID technology (e.g., data striping and mirroring) can help ensure continued operation of a system that suffers hardware or software failure.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Availability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Scalability
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Integrity / Network Infrastructure Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Integrity / Network Infrastructure Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Integrity / Network Infrastructure Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Integrity / Network Infrastructure Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Integrity / Network Infrastructure Integrity
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Integrity / Network Infrastructure Integrity
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Is the system designed to support high availability?

Procedure:

Check for the existence of a cluster and/or failover capability.

Check for the existence of RAID data storage for the database.

Example:

None.

G1357

Do not rely solely on transport level security like SSL or TLS.

Rationale:

Web services inherently involve multiple intermediaries between the message sender and the ultimate destination. The intermediaries may not use transport level security. SSL and TLS do not provide end-to-end security, only security at the transport layer and only point-to-point. The use of SSL or TLS should depend on the needs of the system. For sensitive applications, augment the use of SSL/TLS with defense in depth measures such as message-level security mechanisms.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Mediate Security Assertions
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the Web service user generate encrypted XML messages?

Procedure:

Generate a test message and check it for encryption.

Example:

None.

2) Test:

Does the Web service provider generate encrypted XML messages?

Procedure:

Generate a test message and check it for encryption.

Example:

None.

G1359

Bind SOAP Web service security policy assertions to the service by expressing them in the associated WSDL file.

Rationale:

A Web service may be registered in zero, one, or multiple **UDDI** registries. By placing the security policy assertions in the Web service's WSDL file, they are readily available to all the consumers of the service regardless how the service was discovered

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Mediate Security Assertions
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Are Web service security policy assertions bound in the service WSDL file?

Procedure:

Check the Web Service's WSDL file for policy assertions.

Example:

None

G1362

Validate XML messages against a schema.

Rationale:

Validating messages against a schema helps prevent malicious or malformed data from compromising the integrity of a service. Validating outgoing messages against a schema helps detect compromised services. Validating messages against a schema's data attribution information also enables non-repudiation.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Validate Input

Evaluation Criteria:

1) Test:

Are messages (both incoming and outgoing) validated against a schema?

Procedure:

Identify the existence of an XML Schema file and examine source code to verify that messages are checked against the schema.

Example:

None

G1363

Do not use clear text passwords.

Rationale:

Prevent a hacker from intercepting and seeing a real password.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the Web service user utilize a username/password token?

Procedure:

Generate a test message and check it for clear text passwords.

Example:

None

G1364

Hash all passwords using the combination of a timestamp, a nonce and the password for each message transmission.

Rationale:

This Guidance helps to prevent unwanted interception or discovery of clear-text-hashed passwords.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the Web service user utilize a username/password token?

Procedure:

Generate a test message and check it for a username/password token and verify that is contains a timestamp entry and a nonce entry.

Example:

None

G1365

Specify an expiration value for all security tokens.

Rationale:

Specifying an expiration value for security tokens limits the chance of being able to intercept and use a security token to impersonate an authenticated user or process.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the Web service user utilize an expiration for each security token?

Procedure:

Generate a test message and check it to make sure an expiration is associated with each security token.

Example:

None

G1366

Digitally sign all messages where non-repudiation is required.

Rationale:

Prevent hackers from changing intercepting and modifying a message.

> ***Note:*** *Non-repudiation cannot be assured with soft certificates.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the Web service user digitally sign all messages?

Procedure:

Generate a test message and check it for digital signatures.

Example:

None

2) Test:

Does the Web service provider digitally sign all messages?

Procedure:

Generate a test message and check it for digital signatures.

Example:

None

G1367

Digitally sign message fragments that are required not to change during transport.

Rationale:

Signing message fragments allows the consumer of the message fragment to verify the message fragment has not changed since the producer signed the message fragment.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Do message fragments sent between producers and subscribers have digital signatures when the message content must remain unchanged during transport?

Procedure:

Check system requirements for message fragments that must be transmitted unchanged between the producer and consumer. For these message fragments, check that digital signature are used to detect changes to the message fragments.

Example:

None

G1369

Digitally sign all requests made to a security token service.

Rationale:

Prevent hackers from intercepting a message and requesting a security token.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the Web service user digitally sign all messages?

Procedure:

Generate a test message and check it for digital signatures.

Example:

None

2) Test:

Does the Web service provider digitally sign all messages?

Procedure:

Generate a test message and check it for digital signatures.

Example:

None

G1371

Use the National Institute of Standards and Technology (NIST) *Digital Signature Standard* promulgated in the Federal Information Processing Standards Publication 186 (FIPS Pub 186-3 as of June 2009) for creating Digital Signatures.

Rationale:

Using the FIPS Pub 186-3 ***Digital Signature Standard*** enables interoperability of **Digital Signature Algorithms**.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Cryptography
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the Web service user generate signatures using the FIPS 186-3 ***Digital Signature Standard***?

Procedure:

Generate a test message and check it for compliance with the FIPS 186-3 ***Digital Signature Standard***.

Example:

None

2) Test:

Does the Web service provider generate signatures using the FIPS 186-3 ***Digital Signature Standard***?

Procedure:

Generate a test message and check it for compliance with the FIPS 186-3 ***Digital Signature Standard***.

Example:

None

G1372

Use an X.509 Certificate to pass a Public Key.

Rationale:

This ensures that the owner passing the key is who he says.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the Web service provider send a public key as part of its messages?

Procedure:

Generate a test message and check it for an X.509.

Example:

None

2) Test:

Does the Web service user send a public key as part of its messages?

Procedure:

Generate a test message and check it for an X.509.

Example:

None

G1373

Encrypt messages that cross an IA boundary.

Rationale:

Prevent hackers from reading sensitive information.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the Web service user encrypt all messages?

Procedure:

Generate a test message and check it for encryption.

Example:

None

2) Test:

Does the Web service provider encrypt all messages?

Procedure:

Generate a test message and check it for encryption.

Example:

None

G1374

Individually encrypt sensitive message fragments intended for different intermediaries.

Rationale:

Individually encrypting message fragments allows targeting individual fragments at different intermediaries along the message path to the final destination.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Cryptography
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Are sensitive fragments of the message encrypted?

Procedure:

Observe messages that are sent to see if the sensitive fragments of the message are encrypted.

Example:

None

G1376

Do not encrypt message fragments that are required for correct SOAP processing.

Rationale:

It is possible to encrypt the entire SOAP message, various portions of the SOAP message or the contents of the data transported within the SOAP message. Encrypting the entire SOAP message requires that any intermediate processing of the SOAP message includes decryption of the entire message.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Cryptography
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Other Design Tenets
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Does the Web service user encrypt the entire message?

Procedure:

Generate a test message and check it to make sure the XML tags are not encrypted.

Example:

None

2) Test:

Does the Web service provider encrypt the entire message?

Procedure:

Generate a test message and check it to make sure the XML tags are not encrypted.

Example:

None

G1377

Use LDAP 3.0 or later to perform all connections to LDAP repositories.

Rationale:

Using industry-proven LDAP standards help ensure interoperability of the directory repository with its consumers. LDAP v3 addresses some of the limitations of LDAP v2 in the areas of internationalization and authentication. It also allows adding new features without also requiring changes to the existing protocol through the use of using extensions and controls while maintaining backward compatibility with LDAP v2.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / LDAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / LDAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / LDAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / LDAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / LDAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / LDAP Security
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Check port 636 if supporting secure LDAP (SLDAP)

Procedure:

Test the connection using an SLDAP client.

Example:

None

G1378

Encrypt communication with LDAP repositories.

Rationale:

Encryption of communication to LDAP servers helps prevent disclosure of data during transmission.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / LDAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / LDAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / LDAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / LDAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / LDAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / LDAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Cryptography
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Are connections to LDAP repositories encrypted?

Procedure:

Verify that connections to LDAP repository use Transport Layer Security (TLS) or Secure Sockets Layer (SSL).

Example:

None.

G1379

Use SAML version 2.0 for representing security assertions.

Rationale:

SAML 2.0 supports XML assertions for supporting cross domain access and Web services. The value of this type of access is that the passing of an assertion eliminates the need to create another account in another domain.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Mediate Security Assertions
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Cross-Security-Domains Exchange
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Security Assertion Markup Language (SAML)
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Security Assertion Markup Language (SAML)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Security Assertion Markup Language (SAML)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Security Assertion Markup Language (SAML)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Security Assertion Markup Language (SAML)
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Security Assertion Markup Language (SAML)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Can the SAML message be validated against SAML V2.0 schema?

Procedure:

Validate SAML message against SAML V2.0.

Example:

None.

G1380

Use the XACML 2.0 standard for SAML-based rule engines.

Rationale:

XACML-based rules can define the mechanism for creating the rule and policy set that enable meaningful **authorization** decisions. XAMCL is also integrated with **SAML** to support **role-based access control** or hierarchical resources, such as portions of XML documents.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Mediate Security Assertions
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Cross-Security-Domains Exchange
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Security Assertion Markup Language (SAML)
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Security Assertion Markup Language (SAML)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Security Assertion Markup Language (SAML)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Security Assertion Markup Language (SAML)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Security Assertion Markup Language (SAML)
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Security Assertion Markup Language (SAML)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Does the SAML-based rules engine use the XACML 2.0 standard?

Procedure:

Emulate a rule and run against rule engine using SOAP messaging.

Example:

None.

G1381

Encrypt sensitive persistent data.

Rationale:

When data is persisted, there is always a chance that the security of the system that stores the data may be compromised. To minimize the risk, all sensitive data such as passwords and personal information should be encrypted when it is persisted.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Cryptography
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Cryptography
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Is all sensitive data that is persisted encrypted?

Procedure:

Look at all data stores and check for encrypted passwords and other sensitive data..

Example:

None.

G1382

Be associated with one or more Communities of Interest (COIs).

Rationale:

The DoD Net-Centric Data Strategy emphasizes the establishment of Communities of Interest (**COI**s). This strategy introduces management of data within Communities of Interest (COIs) rather than standardizing **data elements** across the DoD. Thus all DoD Programs must map to one of more COIs. DoD Programs should participate in COIs as a normal course of doing business. They will identity relevant COIs; actively collaborate with them to promote reuse and cross-coordination of **metadata**; sponsor participation of system developers in the COI process and where appropriate contribute engineering expertise to the COI as a stakeholder. New programs should include community collaboration requirements in acquisition documents as required.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Be Responsive to User Needs
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry

NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Is the Program associated with a **COI**?

Procedure:

Check the DoD Metadata registry to determine whether program is associated with any **COI**(s).

Example:

None

G1383

Use a registered namespace in the XML Gallery in the DoD Metadata Registry.

Rationale:

The use of the **DoD Metadata Registry** helps to avoid name collisions and conflicts.

The assignation of a unique **registered namespace** permits a program to be uniquely identified and categorized. The DoD **Net-Centric Data Strategy** requires that data products be stored in shared spaces to provide access to all authorized users and that these data products be tagged with **metadata** to enable discovery of data by authorized users. The use of a unique registered namespace provides an absolute identifier to products associated with a particular product and is an **XSD** schema requirement.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Trustable
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Accessible
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Be Responsive to User Needs
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry

NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Does the Program have an assigned namespace for its XML data assets?

Procedure:

Check the **DoD Metadata Registry** to determine whether the Program is associated with **COI**(s).

Example:

None

G1384

Review XML Information Resources in the DoD Metadata Registry, using those which can be reused.

Rationale:

The DoD Net-Centric Data Strategy requires that **XML** information resources within a **COI** in the **DoD Metadata Registry** be examined by DoD projects for possible reuse to help foster common standards within a **COI** and promote interoperability.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Be Responsive to User Needs
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling

NESI / Part 5: Developer Guidance / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Has the program reused information resources from the **DoD Metadata Registry**?

Procedure:

Check the **XSDs** associated with the program to determine whether XSDs referenced by other namespaces have been used. Check the **DoD Metadata Registry** to determine whether the Program has registered the reuse of XML information resources belonging to other namespaces. Reuse is indicated by formally subscribing to selected components in the registry.

Example:

None

G1385

Identify XML Information Resources for registration in the XML Gallery of the DoD Metadata Registry.

Rationale:

The DoD Net-Centric Data Strategy requires that **XML Information Resources** developed during the course of a program be identified, examined for usefulness by other DoD Programs in the same or related **COIs** and be submitted for inclusion in the XML Gallery of the **DoD Metadata Registry**.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Trustable
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Accessible
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry

NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Has the Program submitted new information resources to the **DoD Metadata Registry**?

Procedure:

Check the **XSDs** associated with the program namespace to determine whether they have been registered in the **DoD Metadata Registry** XML Gallery.

Example:

None

G1386

Review predefined commonly used data elements in the Data Element Gallery of the DoD Metadata Registry, using those in the relational database technology which can be reused in the Program.

Rationale:

The DoD Net-Centric Data Strategy requires that DoD Programs examine data element information resources within a **COI** in the **DoD Metadata Registry** for possible reuse to help foster common standards within a **COI** and promote interoperability. Elements include `US State Codes` and `Country Codes`. This reuse is preferential to reusing existing industry standard **data elements** or developing new **data elements**.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Be Responsive to User Needs
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry

NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Has the Program reused common database elements?

Procedure:

Check the DoD Metadata Registry Data Element Gallery to determine whether the program has registered database elements for reuse. Reuse is indicated by formally subscribing to selected components in the registry.

Check the program database to see whether registered have been included therein.

Example:

None

G1387

Identify data elements created during Program development for registering in the Data Element Gallery of the DoD Metadata Registry.

Rationale:

The DoD Net-Centric Data Strategy requires that Programs identify and examine developed **data elements** for usefulness by other DoD Programs in the same or related **COIs** and submit the data elements for inclusion in the **Data Element Gallery** of the **DoD Metadata Registry**.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Trustable
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Accessible
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Has the Program submitted common database elements to the **DoD Metadata Registry**?

Procedure:

Check the DoD Metadata Registry Data Element Gallery to determine whether the program has submitted database elements for reuse.

Example:

None

G1388

Use predefined commonly used database tables in the DoD Metadata Registry.

Rationale:

The *DoD Net-Centric Data Strategy* [R1172] requires that DoD Programs examine data table information resources within a **COI** in the **DoD Metadata Registry** for possible reuse to help foster common standards within a COI and promote interoperability. This reuse is preferable to reusing existing industry standard **data elements** or developing new data elements. Some examples are **`Country Code`**, **`US State Code`**, **`Purchase Order Type Code`**, **`Security Classification Code`**. These tables are found in the **Reference Data Set** Gallery of the DoD Metadata Registry.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Trustable
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Be Responsive to User Needs
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry

NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Has the Program reused common database tables?

Procedure:

Check the DoD Metadata Registry to determine whether the program has registered database tables for reuse. Reuse is indicated by formally subscribing to selected components in the registry.

Check the program database to see whether registered data tables have been included therein.

Example:

None

G1389

Publish database tables which are of common interest by registering them in the Reference Data Set Gallery of the DoD Metadata Registry.

Rationale:

The DoD Net-Centric Data Strategy requires that DoD Programs identify and examine developed data tables for usefulness by other DoD Programs in the same or related **COIs** and be submit the data elements for inclusion in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Trustable
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Accessible
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Be Responsive to User Needs
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Has the Program submitted common database tables to the DoD Metadata Registry?

Procedure:

Check the DoD Metadata Registry Reference Data Set Gallery to determine whether the program has submitted database tables for reuse.

Example:

None

G1391

Identify taxonomy additions or changes in conjunction with the Communities of Interest (COIs) during the Program development for potential inclusion in the Taxonomy Gallery of the DoD Metadata Registry.

Rationale:

DoD Programs associated with a specific COI need to identify and submit potential taxonomy changes or additions to the **DoD Metadata Registry** to maintain an accurate and effective taxonomy within the **COI**.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Accessible
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Be Responsive to User Needs
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Has the Program submitted **taxonomy** additions or changes to the **DoD Metadata Registry**?

Procedure:

Check the DoD Metadata Registry and to determine whether the program has submitted taxonomy changes for reuse.

Example:

None

G1566

Use `alt` attributes to provide alternate text for non-text items such as images.

Rationale:

This usage aids users in understanding the Web page even if their browsers cannot display images.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Are alt attributes provided for non-text content?

Procedure:

Check for the existence of alt attributes for all Web site non-text content.

Example:

None.

G1713

Use an Operating Environment (OE) for all Software Communications Architecture (SCA) applications that includes middleware which adheres to the Minimum CORBA Specification version 1.0.

Rationale:

Using a CORBA provider that adheres to the minimum CORBA v1.0, specification improves the interoperability between SCA Operating Environments.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: RF Acquisition
NESI / Part 2: Traceability / DISR Service Areas / Communications Applications / Software Communication Architecture
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Software Communication Architecture
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Communication Architecture
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Software Communication Architecture
NESI / Part 5: Developer Guidance / Middleware / Software Communication Architecture
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the OE contain middleware that provides the services and capabilities of minimum CORBA?

Procedure:

Check for minimum CORBA compliance in the CORBA provider's documentation.

Example:

None.

G1714

Develop Software Communications Architecture (SCA) applications to use only Operating Environment functionality defined by the SCA Application Environment Profile.

Rationale:

The SCA Application Environment Profile (AEP) is a subset of the Portable Operating System Interface (POSIX) specification. Functionality that is not part of the AEP is not guaranteed to be part of the operating environment. Applications that rely on functionality that is not part of the AEP will require changes to deploy or port to other SCA platforms.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: RF Acquisition
NESI / Part 2: Traceability / DISR Service Areas / Communications Applications / Software Communication Architecture
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Software Communication Architecture
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Communication Architecture
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Software Communication Architecture
NESI / Part 5: Developer Guidance / Middleware / Software Communication Architecture
NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the SCA application use Operating Environment functions not defined by a Application Environment Profile?

Procedure:

Check to see that all Operating Environment calls in the SCA application are listed in an Application Environment Profile.

Example:

None.

G1717

Use constants instead of hard-coded numbers for characteristics that may change throughout the lifetime of the model.

Rationale:

Constants increase the usefulness and lifetime of a design because the model can adapt to a variety of environments by postponing or modifying those parameters late in the design cycle. This makes the code more readable, maintainable and reusable.

> ***Note:*** *This practice has been adapted from Cohen [R1114], section 1.6.1.1.3.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / VHDL / VHDL Coding and Design
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability

Evaluation Criteria:

1) Test:

Are there any characteristics that are susceptible to modification that are directly given a value?

Procedure:

Parse the code and look for hard-coded characteristics that are susceptible to change and consider replacing them with a constant.

Example:

None

G1718

Design circuits to be synchronous.

Rationale:

The preferred method of engineering today's digital ICs is based on a synchronous design. The main advantages of this are simplicity and reliability. Creating synchronous pieces of code increases interoperability and reusability when they are used with other synchronous modules.

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / VHDL / VHDL Synchronous Design
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability

Evaluation Criteria:

1) Test:

Are all flip-flops clocked by the same, common clock signal?

Procedure:

Check to make sure a single external clock signal triggers the design to go from a well defined and stable state to the next one. On the active edge of the clock, all input and output signals and all internal nodes are stable in either the high or low state. Between two consecutive edges of the clock, the signals and nodes are allowed to change and may take any intermediate state.

Example:

None

G1719

Automate testbench error checking in VHDL development.

Rationale:

Manual verification is subject to human error and is time consuming. In addition, automation promotes increased maintainability, because it enables fast and reliable verification of a model when modifications are made.

> ***Note:*** *This practice has been adapted from Cohen [R1114], section 11.1.1.*

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Composeability
NESI / Part 5: Developer Guidance / Programming Languages / VHDL / VHDL Testbench
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability

Evaluation Criteria:

1) Test:

Does the testbench automatically report success or failure for each sub-test that it runs through?

Procedure:

Run the testbench to see if it automatically reports successes or failures for each sub-test.

Example:

None

G1724

Develop XML documents to be well formed.

Rationale:

By **W3C** definition, **XML documents** must be **well formed**. However, documents that contain XML tags that are not well formed has no name and is often still referred to as an XML Document in common vernacular. Therefore, this guidance statements helps to clarify the need for well-formed documents. Well formed XML documents are those documents which have a proper XML syntax. This is essential if the XML is to be parsed using common, readily available open source and commercial XML parsers.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Syntax
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Syntax
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Syntax
NESI / Part 5: Developer Guidance / Data / XML / XML Syntax
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Can the XML Document be parsed using a common, readily available XML Parser?

Procedure:

Open the XML document in a browser such as Mozilla Firefox or Microsoft Internet Explorer or use the XML Validator available from the W3 Schools at http://www.w3schools.com/xml/xml_validator.asp.

Example:

None.

G1725

Develop XML documents to be valid XML.

Rationale:

The content of a **valid** XML document conforms to a specific set of user-defined content rules contained in XML schemas. XML schemas describe data values correctness using predefined data types as base types and assigning values to the data type specific attributes of those data types. For example, if an element in a document is required to contain text that can be interpreted as being an integer numeric value, and instead contains: alphanumeric text such as "hello"; is empty; or has other elements in its content, then the document is considered not valid.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / XML Validation

NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / XML Validation
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / XML Validation
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / XML Validation
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the document validation tool indicate that the XML document is valid?

Procedure:

Use a validating parser and verify that the document is valid.

Example:

None.

G1726

Define XML Schemas using XML Schema Definition (XSD).

Rationale:

While it is possible to use **Document Type Definitions** (**DTD**) to convey much of the same information as the **XML Schema Definition** (**XSD**), XSDs have a several distinct advantages which are very useful in terms of interoperability. For example, DTDs do not capture domain or type range information very well (i.e., elevation in meters is from 0 to 12,000).

XML Schemas are a tremendous advancement over DTDs. Here are some of the reasons to use XSDs versus DTDs as delineated by Roger Costello in an XML tutorial (see the ***XML Schema Tutorial*** available at http://www.xfront.com):

- Enhanced datatypes support:
  - 44+ in XSDs versus 10 in DTDs
  - Support for user defined datatypes. For example, a user can define a new type based on the string type. Elements declared of this type must follow this specific pattern ddd-dddd, where d represents a numeric digit.
- Written using the same syntax as other XML instance documents. This means there is less to remember and more consistency with the same rules applying to all XML instance documents.
  XSDs support a limited Object-oriented (OO) paradigm. For example, new types can be derived from previously defined types with more or more stringent restrictions.
- Supports a kind of polymorphism where elements can be interchanged with parent or child elements. For example, a "Book" element can be substituted for the "Publication" element.
- Supports the definition of elements that are unordered collections or sets of other elements.
- Support for the identification of elements as part of a unique key.
- Support for elements that have the same name but different content
- Support for elements that have a null (i.e., nil) value.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management

NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Are XML schemas defined using XML Schema Definitions?

Procedure:

Verify that XML schemas are defined using W3C XML Schema Definitions rather than Document Type Definitions.

Example:

None.

G1727

Provide names for XML type definitions.

Rationale:

By naming type definitions in a schema, the type definitions can be reused in any number of other definitions. For example:

```
<xsd:complexType name="PointOfContact">
 <xsd:sequence>
   <xsd:element name="LastName" type="xsd:string"/>
   <xsd:element name="FirstName" type="xsd:string"/>
   <xsd:element name="MiddleName" type="xsd:string"/>
   <xsd:element name="NickName" type="xsd:string"/>
   <xsd:element name="PhoneNumber" type="xsd:string"/>
 </xsd:sequence>
</xsd:complexType>
```

Can be reused anywhere a Point-Of-Contact needs to used. For Example:

```
<xsd:complexType name="Project">
 <xsd:sequence>
   <xsd:element name="ProjectName" type="xsd:string"/>
   <xsd:element name="ProgramManager" type="PointOfContact"/>
   <xsd:element name="HardwareManager" type="PointOfContact"/>
   <xsd:element name="SoftwareManager" type="PointOfContact"/>
   <xsd:element name="ConfigurationManager" type="PointOfContact"/>
 </xsd:sequence>
</xsd:complexType>
```

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Versioning XML Schemas

NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Do all simpleTypes have names associated with them?

Procedure:

Examine all the simpleType elements in the schema and verify that they have a name associated with them.

Example:

```
<xsd:simpleType name="PointOfContact">
   ...
</xsd:simpleType>
```

2) Test:

Do all complexTypes have names associated with them?

Procedure:

Examine all the complexType elements in the schema and verify that they have a name associated with them.

Example:

```
<xsd:complexType name="PointOfContact">
  ...
</xsd:complexType>
```

G1728

Define types for all XML elements.

Rationale:

There are two ways to associate the type-like information within an XML Schema. The first way is define an **XML element** as a global element of the schema element and the second is to define a complex or simple type. The first method violates G1727 and it does not support the clean separation of the definition of types from the use of the types.

By separating the definition of the types from the definition of the elements within structures, the types can be reused and are loosely coupled from any particular instance of the domain. The definitions of the type information can be maintained by a community that wishes to share the definition rather than any particular implementation or instance.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the schema define any elements that are defined using references to other elements that are not part of a substitutionGroup rather than types?

Procedure:

Look for the use of an element's ref attribute.

Example:

None.

G1729

Annotate XML type definitions.

Rationale:

Types in a schema represent a particular concept or aspect within a particular subject domain. Providing documentation about the type within the schema itself helps prevent disconnects between the documentation and the implementation as captured by the type definition.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Do all the types defined within a schema have annotation that describes the nuances of type?

Procedure:

Look for an annotation for each simple type and complex type defined in the schema.

Example:

The complex type warranty includes an annotation that describes the purpose of the type and any caveats on when/how to use it.

G1730

Follow a documented XML coding standard for defining schemas.

Rationale:

There are any number of coding conventions that are defined for coding XML Schemas. Here are some areas covered by the most popular:

- Elements and Types are **Upper Camel Case** (**UCC**) convention.
- Type names end with the word Type.
- Attributes start with a lowercase letter and then revert to **Lower Camel Case** (**LCC**) convention.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Is there a consistent XML coding convention followed when schemas are defined?

Procedure:

Look for the occurrence of a XML coding standard and verify that the XML Schemas follow the standard.

Example:

None.

G1731

Only reference XML elements defined by a Type in substitution groups.

Rationale:

The 35mm, disk, and 3x5 components are simply declared as standalone **XML elements** which may be substituted for the abstract **`RecordingMedium`** element.

> ***Note:*** *All of these* ***`RecordingMedium`*** *components have a type that is the same as, or derived from, the* ***`RecordingMedium`****Type.*

> ***Note:*** *The abstract* ***`RecordingMedium`*** *is associated with a type,* ***`RecordingMediumType`****, rather than defining the structure as part of the* ***`RecordingMedium`*** *element. This allows the definition of the* ***`RecordingMedium`*** *structure (i.e., type) to evolve independently.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability

Evaluation Criteria:

1) Test:

Do **`substitutionGroup`** references point to an abstract element that has a structures defined by a type?

Procedure:

Ensure that all **`substitutionGroups`** point to an abstract element that has a structures defined by a type.

Example:

None.

G1735

Use the `.xsd` file extension for files that contain XML Schema definitions.

Rationale:

It is possible to use any name for a schema file extension. However, using any extension other than **`.xsd`** causes confusion for humans as well as tools and utilities which rely on MIMEs often mapped to file extensions.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability

Evaluation Criteria:

1) Test:

Is the file extension that contains the schema definition .xsd?

Procedure:

Make sure that all XML documents that contain the xml **`schema`** tag have a file extension of .xsd.

Example:

None.

G1736

Separate document schema definition and document instance into separate documents.

Rationale:

Separating the definition of the schema from the document instance supports the modularity by separating the definition of structure from the actual data. Each is allowed to evolve and change independently. In most cases, the definition of the structure of the data should be relatively static compared with the number of documents that are shared using that schema.

Document name: Camera.xsd

```
<xsd:schema
    targetNamespace="http://www.camera.org"
    elementFormDefault="qualified">
 <xsd:include schemaLocation="Nikon.xsd"/>
 <xsd:include schemaLocation="Olympus.xsd"/>
 <xsd:include schemaLocation="Pentax.xsd"/>
 <xsd:element name="Camera">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element
          name="Body"
          type="BodyType"/>
       <xsd:element
          name="Lens"
          type="LensType"/>
       <xsd:element
          name="ManualAdapter"
          type="ManualAdapterType"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Document name: Camera.xml

```
<?xml version="1.0"?>
<Camera xmlns ="http://www.camera.org"

        xsi:schemaLocation=
                   "http://www.camera.org
                    Camera.xsd">
 <Body>
   <Description>
      Ergonomically designed casing for easy handling
   </ Description>
 </Body>
 <Lens>
   <Zoom>300mm</Zoom>
   <F-Stop>1.2</F-Stop>
 </Lens>
 <ManualAdapter>
   <speed>1/10,000 sec to 100 sec</speed>
 </ManualAdapter>
</Camera>
```

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / XML Schema Files

NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / XML Schema Files
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability

Evaluation Criteria:

1) Test:

Does the instance document have a <schema> tag?

Procedure:

Check the instance document and look for the use of the schema tag or the use of the XMLSchema namespace.

Example:

None.

G1737

Define a target namespace in schemas.

Rationale:

A target namespace describes the namespace for all the schema components defined by the schema. Without a target namespace, all enclosed schema components are not associated with a namespace and if a namespace prefix is not associated with the target namespace then all references to these schema components must be unqualified. By not specifying a target namespace, ambiguity can arise when the schema is integrated with other schemas. This can cause unnecessary naming collisions.

> ***Note:*** *http://www.library.org is the target namespace as well the lib namespace. See the third targetNamespace line of the following code sample.*

```
<?xml version="1.0"?>
<xsd:schema
      targetNamespace="http://www.library.org"

      elementFormDefault="qualified">
 <xsd:include schemaLocation="BookCatalogue.xsd"/>
 <xsd:element name="Library">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="BookCatalogue">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element ref="lib:Book"
                          maxOccurs="unbounded"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the schema declare a target namespace?

Procedure:

Check the definition of all schemas and look for the assignment of the targetNamespace attribute.

Example:

```
<xsd:schema

  targetNamespace="http://www.library.org"
 >
  . . .
</xsd:schema>
```

G1738

Define a qualified namespace for the target namespace.

Rationale:

To force all schema components defined by the schema to be qualified and to belong to a namespace, associate a qualified namespace with the target namespace. This causes all components defined within the namespace to be explicitly associated with a namespace. In other words, all components are always qualified.

> ***Note:*** *http://www.library.org is the target namespace as well the lib namespace. See the forth xmlns:lib line of the following code sample.*

```
<?xml version="1.0"?>
<xsd:schema
      targetNamespace="http://www.library.org"

      elementFormDefault="qualified">
 <xsd:include schemaLocation="BookCatalogue.xsd"/>
 <xsd:element name="Library">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="BookCatalogue">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element ref="lib:Book"
                           maxOccurs="unbounded"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable

NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the schema declare a qualified namespace for the target namespace?

Procedure:

Check the definition of all schemas and look for the assignment of the targetNamespace attribute and make sure there is also a qualified namespace with the same name.

Example:

In this example, the targetNamespace and the qualified namespace lib both have the same URI associated with them.

```
<xsd:schema

  targetNamespace="http://www.library.org"
 >
  . . .
</xsd:schema>
```

G1740

Append the suffix Type to XML type names.

Rationale:

Syntactically, XML allows names within a namespace to be reused as long as they do not define the same XML Schema component. Therefore, a type and an element can both have the same name. A parser can easily differentiate the components, but a human can not. In order to maintain maintainable "user-friendly" code, differentiate types and elements by adding a type suffix for types.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability

Evaluation Criteria:

1) Test:

Do all the complex type names end in the type suffix?

Procedure:

Examine all the complex and simple type schema component definitions and verify that they end in the suffix type.

Example:

None.

G1744

Only reference abstract XML elements in substitution groups.

Rationale:

An abstract **XML element** can not have its type instantiated in an instance document. This means that the element used as the basis for the substitution group and all the members of the substitution group must be derived from the same type.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability

Evaluation Criteria:

1) Test:

Is the element used as the basis for the substitution group declared to be abstract and is it derived from a type?

Procedure:

Examine all the elements used as the basis for substitution groups and verify that they have been declared as abstract.

Example:

```
<xsd:element name="RecordingMedium"
      abstract="true"
      type="RecordingMediumType"/>
```

G1745

Append the suffix Group to substitution group XML element names.

Rationale:

Syntactically, XML allows names within a namespace to be reused as long as they do not define the same XML Schema component. Therefore, a type and an **XML element** can both have the same name. A parser can easily differentiate the components, but a human can not. In order to maintain maintainable "user-friendly" code, differentiate types and elements by adding a type suffix for types.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Using XML Substitution Groups
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability

Evaluation Criteria:

1) Test:

Do all the complex type names end in the type suffix?

Procedure:

Examine all the complex and simple type schema component definitions and verify that they end in the suffix type.

Example:

None.

G1746

Develop XSLT style sheets that are XSLT version agnostic.

Rationale:

There are never any guarantees as to the XSLT environment that a stylesheet will be used in. There are ways of writing code as recommended by the W3C so that the stylesheets operate in XSL Version 1.0, 2.0 and future releases. See W3C Extensibility and Fallback for XSL Transformations (XSLT) 2.0 for details.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / XSLT
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the style sheet support version 1.0 and 2.0 portability as defined by the W3C Extensibility and Fallback for XSL Transformations (XSLT) 2.0?

Procedure:

Look for the use of the xsl:when and xsl:otherwise construct where the 2.0 functions are tested for availability in the xsl:when branch and the 1.0 functionality is defined in the xsl:otherwise branch. For a comprehensive list of 2.0 functions see the W3Schools site on XPath, XQuery and XSLT Functions.

Example:

```
<out xsl:version="2.0">
 <xsl:choose>
   <xsl:when
      test="function-available('matches')">
     <xsl:value-of
        select="matches($input, '[a-z]*')"/>
   </xsl:when>
   <xsl:otherwise>
     <xsl:value-of
        select=
          = "string-length
              ( translate
               ( $in,
                 'abcdefghijklmnopqrstuvwxyz',
                   ''
               )
              )
             = 0"
      />
   </xsl:otherwise>
 </xsl:choose>
</out>
```

2) Test:

Does the style sheet support 2.0 and future version portability as defined by the W3C Extensibility and Fallback for XSL Transformations (XSLT) 2.0?

Procedure:

Look for the use of the use-when attribute in the xsl:value element.

Example:

```
<xsl:value-of
   select="pad($input, 10)"
   use-when="function-available('pad', 2)"
/>
<xsl:value-of
  select
    ="concat
       ( $input,
         string-join
          ( for $i in
              1 to
              10 - string-length($input)
              return ' ',
           ''
          )
       )"
  use-when="not(function-available('pad', 2)
"/>
```

G1751

Document all XSLT code.

Rationale:

XSLT is source code and should be internally documented including a file header that describes the purpose of the transform and any restrictions or caveats associated with the transform.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / XSLT
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability

Evaluation Criteria:

1) Test:

Doe the XSLT have internal comments that document the transform?

Procedure:

Look inside the XSLT code and look for internal comments.

Example:

```
<xsl:for-each
  select="/transactions/transaction">
 <!--
    NOTE: Since dates are currently in
    ISO format they are in a sorted format
    and need no multi-level sorting
  -->
 <xsl:sort
    order="ascending"
    select="@startdate"/>
 <tr>
   <td>
     <xsl:value-of
        select="@startdate"/>
   </td>
   <td>
     <xsl:value-of
        select="@description"/>
   </td>
   <td>
     <!#  Get year
           1234567890
           yyyy/mm/dd
      -->
     <xsl:value-of
        select="substring(@startdate, 1,4)"
      />
   </td>
   <td>
     <!#  Get month
           1234567890
           yyyy/mm/dd
      -->
     <xsl:value-of
        select="substring(@startdate, 6,2)"/>
```

```
    </td>
    <td>
      <!#  Get day
            1234567890
            yyyy/mm/dd
       -->
      <xsl:value-of
         select="substring(@startdate, 9,2)"/>
    </td>
  </tr>
</xsl:for-each>
```

G1753

Declare the XML schema version with an XML attribute in the root XML element of the schema definition.

Rationale:

Formalizing the schema version number through the use of a required **XML attribute** helps automate the process of validating the versions. This will reduce unexpected runtime errors that occur when assumptions are made about the schema that may change over time. (See http://www.xfront.com/SchemaVersioning.html)

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Does the schema definition define a required attribute that captures the version information?

Procedure:

Look at the schema definition file and look for the inclusion of a required attribute that captures the schema version number. In the following example, the schemaVersion attribute is defined.

Example:

```
<xs:schema
```

```
  targetNamespace="http://www.exampleSchema"
  xmlns: xs ="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="1.3"
>
 <xs:element name="Example">
   <xs:complexType>
      . . .
     <xs:attribute
        name="schemaVersion"
        type="xs:decimal"
        use="required"
      />
   </xs:complexType>
 </xs:element>
```

G1754

Give each new XML schema version a unique URL.

Rationale:

This allows the previous versions of the schema to be made available to support uninterrupted processing and supports an orderly transition. It also allows the users of the schemas to compare and contrast the evolving schema. http://www.xfront.com/SchemaVersioning.html

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Versioning XML Schemas
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Look for the multiple schemas that represent different versions with different URLs.

Procedure:

Look for XSDs that all define a particular schema but can be found at different locations. This can be done by changing the path to the schema definition or that change the name of the file by adding the version number.

Example:

Changing the file path:

```
http://www.some.org/schema/1999/CoiSchema
http://www.some.org/schema/2003/CoiSchema
http://www.some.org/schema/2006/CoiSchema
```

Changing the file name:

```
http://www.some.org/schema/CoiSchema_1999
http://www.some.org/schema/CoiSchema_2003
http://www.some.org/schema/CoiSchema_2006
```

G1755

Use accepted file extensions for all files that contain XSL code.

Rationale:

It is possible to use any name for an XSL file extension. However, using any extension other than xsl or XSLT causes confusion for humans as well as tools and utilities which rely on MIMEs often mapped to file extensions.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / XSLT
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability

Evaluation Criteria:

1) Test:

Is the file extension that contains the XSL files .xsl or .xslt?

Procedure:

Make sure that all XSL files have a file extension of .xsl or xslt.

Example:

None.

G1756

Isolate XPath expression statements into the configuration data.

Rationale:

XPath expression statements are dependent on the XML Schemas that are associated with the documents. Consequently they need maintained independently from the applications that use them. Storing the XPath expression statements externally as part of the configuration data ensures a clean separation of the maintenance tasks and supports traceability using configuration management tools.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / XPath
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / XPath
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / XPath
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / XPath
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability

Evaluation Criteria:

1) Test:

Are there XPath expression statements embedded as string literals in the application source code?

Procedure:

Look for the occurrence of XPath expression statements or XML Element names defined as strings within the source code.

Example:

```
void main ( String args)
{ . . .
  String titleSearchExpression
    = "/library/books/book/title";
  . . .
} // End main
```

G1759

Use a style guide when developing Web portlets.

Rationale:

Portals contain portlets from different sources, and it is important for usability for the portal to have a common look and feel across all portlets.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces

Evaluation Criteria:

1) Test:

Do all portlets comply with a style guide.

Procedure:

Look at development documentation to determine if a style guide exist for Web portlets and look for code reviews that show it was used during development.

Example:

None.

G1760

Solicit feedback from users on user interface usability problems.

Rationale:

Active testing and solicitation of input from users helps identify usability problems with the user interface and helps to identify areas that may reduce performance or require excessive cognitive attention by the user.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Be Responsive to User Needs
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Does the program solicit user feedback for user interface usability problems?

Procedure:

Determine if user surveys are conducted on the usability of the system.

Example:

None.

G1761

Provide units of measurements when displaying data.

Rationale:

Displayed units for measurable data provide for better understanding the data and enable reuse of the data. (This guidance is derived from MIL-STD-1472F.)

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Does the system display units for all measurable data?

Procedure:

Inspect the user interfaces for system and check that units are shown for all measurable data.

Example:

1. Length displayed as meters.
2. Distance displayed as miles.

G1762

Indicate all simulated data as simulated.

Rationale:

Simulated data that is not marked as simulated may be of misinterpreted and can decrease system, user, or system safety. (This guidance is derived from MIL-STD-1472F.)

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Trustable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Is all simulated data clearly marked as simulated?

Procedure:

Check system inputs and outputs including user interfaces and check that the simulated data is properly labeled as simulated.

Example:

None.

G1763

Indicate the security classification for all classified data.

Rationale:

Displaying classified data without clearing marking the classification can lead to incorrect assumptions about the data. This can lead to improperly use of the data or prevent the data from being reused due to lack of clear understanding of the classification. (This guidance is derived from MIL-STD-1472F.)

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Trustable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Accessible
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Does the system display classification markings for all classified data?

Procedure:

Check the system outputs and user interfaces for classification marking for all classified data.

Example:

Classification banners on monitors and printouts.

G1770

Explicitly define Data Distribution Service (DDS) Domains.

Rationale:

DDS uses Domains to separate the **Global Data Spaces** into independent areas. **Topics** written to one DDS Domain are completely hidden from the other DDS Domains. Use DDS Domains for isolation (hiding subsystem data from other parts of the system), modularity, and scalability. In order for systems to benefit from these advantages, they must explicitly define their own DDS Domains rather than use the default DDS Domain.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture

Evaluation Criteria:

1) Test:

Is the system using different **`DomainId`** values to isolate the subsystems?

Procedure:

Look for multiple calls to **`create_participant()`** operation on the **`DomainParticipantFactory`**.

Example:

```
participantFactory
  = TheParticipantFactory;
quickQuoterParticipant
  = participantFactory->create_participant
      ( QUICK_QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
realtimeQuoterParticipant
  = participantFactory->create_participant
      ( REALTIME_QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
```

```
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111

G1771

Explicitly define the Data Distribution Service (DDS) Quality of Service (QoS) Policies to describe the behavior of a publisher.

Rationale:

DDS relies on the use of QoS characteristics to match publishers with **subscribers**. If the publishers do not specify a QoS policy other than the default, much of the power of DDS publishing is lost and the capabilities of the publisher are not documented.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Differentiated Management of Quality-of-Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Is the **`get_default_publisher_qos`** operation used to create publisher?

Procedure:

Look for the use of the **`get_default_publisher_qos`** operation within the code.

Example:

```
participant
  = participantFactory->create_participant
      ( QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
DDS::PublisherQos publisherQos;
Participant->get_default_publisher_qos
  ( publisherQos );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111

2) Test:

Are values other than the **`PUBLISHER_QOS_DEFAULT`** value used to create publishers?

Procedure:

Verify that the **`PUBLISHER_QOS_DEFAULT`** constant is not used within the code.

Example:

```
DDS::Publisher publisher
  = participant->create_publisher
      ( PUBLISHER_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111

G1772

Assign a unique identifier for each Data-Distribution Service (DDS) Domain.

Rationale:

DDS uses Domains to separate the **Global Data Spaces** into independent areas. Within DDS, a unique identifier called the **`DomainId`** identifies each DDS Domain.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / DDS Domains - Global Data Spaces
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Is there a single value for the **`DomainId`** used for each Domain when the **`create_participant`** operation is used?

Procedure:

Look for the use of the **`create_participant`** operation within the code.

Example:

```
participantFactory
  = TheParticipantFactory;
quickQuoterParticipant
  = participantFactory->create_participant
      ( QUICK_QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
realtimeQuoterParticipant
  = participantFactory->create_participant
      ( REALTIME_QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111

G1773

Use `#include` guards for all headers.

Rationale:

Including a guard prevents including a header file more than once. There are two basic kinds of guards: internal and external. Internal guards occur in each header file that is to be included. External guards occur in a file that includes a header file. In the past, there were compiling performance issues using internal guards because the file had to be scanned each time the file was included. This has been optimized away by most modern compilers. Furthermore, external guards are fragile and tightly coupled since the file including the header and header file must use the same guard name.

> ***Note:*** *This practice has been adapted from Sutter and Alexandrescu [R1150], standard practice 24.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / C++ / C++ Header Files
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability

Evaluation Criteria:

1) Test:

Do all header files contain include guards?

Procedure:

Check each file that is included using a **`#include`** statement to make sure it has an include guard.

Example:

An internal guard looks like this:

```
#ifndef MYHEADER_HPP

#define MYHEADER_HPP

... // Contents of include file go here

#endif
```

G1774

Make header files self-sufficient.

Rationale:

To enable code reuse, each unit of code should be able to be compiled independently without having to follow a predetermined build order or having to know the dependencies. Code is difficult to reuse when the dependencies are not clearly documented. Therefore, ensure each header is capable of being used by itself (i.e., it can be compiled standalone) by having it include all the headers upon which it depends.

> ***Note:*** *This practice has been adapted from Sutter and Alexandrescu [R1150], standard practice 23.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / C++ / C++ Header Files
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability

Evaluation Criteria:

1) Test:

Can each class be compiled by itself without having to compile other units?

Procedure:

Compile each class as a standalone file and check compile output for errors caused by missing definitions.

Example:

None

G1775

Do not overload the logical `AND` operator.

Rationale:

The logical **`AND`** operator has a special relationship with the compiler. When a logical **`AND`** operator is written to overload the inherent operators, the precedence of operation (i.e., left side of operator or right side of operator) is undefined. This can result in compiler dependency. In the following code, it is not clear whether the **`DisplayPrompt`** will execute first or the **`GetLine`** operation will executed first.

```
if ( DisplyPrompt() && GetLine() )
```

> ***Note:*** *This practice has been adapted from Sutter and Alexandrescu [R1150], standard practice 30.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / C++ / C++ Operator Overloading
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability

Evaluation Criteria:

1) Test:

Is the logical **`AND`** operator defined?

Procedure:

Look for the overloading of the logical **`AND`** operator.

Example:

None

G1776

Do not overload the logical `OR` operator.

Rationale:

The logical `OR` operator has a special relationship with the compiler. When a logical `OR` operator is written to overload the inherent operators, the precedence of operation (i.e., left side of operator or right side of operator) is undefined. This can result in compiler dependency.

> ***Note:*** *This practice has been adapted from Sutter and Alexandrescu [R1150], standard practice 30.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / C++ / C++ Operator Overloading
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability

Evaluation Criteria:

1) Test:

Is the logical `OR` operator defined?

Procedure:

Look for the overloading of the logical `OR` operator.

Example:

None

G1777

Do not overload the `comma` operator.

Rationale:

The **`comma`** operator has a special relationship with the compiler. When a **`comma`** operator is written to overload the inherent operators, the precedence of operation (i.e., left side of operator or right side of operator) is undefined. This can result in compiler dependency.

> ***Note:*** *This practice has been adapted from Sutter and Alexandrescu [R1150], standard practice 30.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / C++ / C++ Operator Overloading
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability

Evaluation Criteria:

1) Test:

Is the **`comma`** operator defined?

Procedure:

Look for the overloading of the **`comma`** operator.

Example:

None

G1778

Place all `#include` statements before all namespace `using` statements.

Rationale:

Files that are included can contain their own **`using`** clauses. In order to make sure that the **`using`** statements are not overridden by these subsequent using definitions, place all using statements after all include statements.

> ***Note:*** *This practice has been adapted from Sutter and Alexandrescu [R1150], standard practice 59.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / C++ / C++ Namespaces and Modules
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability

Evaluation Criteria:

1) Test:

Are all the **`using`** statements defined after all the **`#include`** statements?

Procedure:

Scan all files and make sure that all the **`using`** statements occur after all **`#include`** statements.

Example:

None

G1779

Explicitly namespace-qualify all names in header files.

Rationale:

Header files are meant to be included by other files. A header file inclusion should not alter the meaning of code that it is included in as this behavior is unexpected. Therefore, use fully-qualified names in header files and do not use using directives or declarations. This also promotes clarity in the header file whose main purpose is to communicate the interface to the implementation class.

> ***Note:*** *This practice has been adapted from Sutter and Alexandrescu [R1150], standard practice 59.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / C++ / C++ Namespaces and Modules
NESI / Part 5: Developer Guidance / Programming Languages / C++ / C++ Header Files
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Reusability

Evaluation Criteria:

1) Test:

Are named fully namespace qualified throughout the header files?

Procedure:

Scan all header files and make sure that all namespaces are fully qualified.

Example:

None

2) Test:

Are all header files free from using directives or declarations?

Procedure:

Scan all header files to determine that they do not contain using directives or declarations.

Example:

None

G1796

Explicitly define Data Distribution Service (DDS) Domain Topics.

Rationale:

DDS uses Topics to define the information model. Topics are identified by an application-defined string and an associated **data type**. Topics represent collections of object sin the **Global Data Space**; individual data-objects within a Topic are identified by the value of the key fields which are some special fields inside the data-type. Applications use Topics to publish the information and subscribe to the information they want.

In a DDS system information exchange happens as a result of **publishers** and **subscribers** agreeing to use the same Topics. Therefore, the selection of the Topic names and their semantic meaning is an important part of system design.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Are all the Topics (and Topic names) explicitly defined and captured in a publicized data source (e.g., Excel table, XML file, dedicated tool)?

Procedure:

Look for documentation that contains listings for all Topics the system uses.

Example:

```
<topic>
 <name>Temperature</name>
 <type>TemperatureData</type>
 <description>
    This topic contains a reading of
    a temperature sensor
 </description>
</topic>
<topic>
```

```
  . . .
</topic>
```

G1797

Use a minimum of 1024 bits for asymmetric keys.

Rationale:

Strong encryption helps to prevent unauthorized data decryption using modern day resources.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Encryption Services
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Are asymmetric key encryption levels at least 1024 bit?

Procedure:

Check the server configuration and verify that the asymmetric keys being used are at least 1024 bit.

Example:

Verified Web server ciphers under the SSL portion of the configuration pages of the administration server.
For Internet Explorer 5.0 and above, click the **`Help`** menu and then click the **`About Internet Explorer`** option. The **`About`** box will list the Cipher Strength.

2) Test:

Is the application using domestic (U.S.) grade ciphers?

Procedure:

Verify that the application supports domestic (U.S.) grade ciphers.

Example:

None.

G1798

Explicitly define all the Data Distribution Service (DDS) Domain data types.

Rationale:

DDS provides support for writing and reading typed data. For each application data type, DDS creates the necessary objects that allow manipulation of the data object. For example, for a given data type named **`MyDT`**, DDS creates a **`MyDTDataWriter`** and **`MyDTDataReader`**.

Knowing the data type of the object allows DDS to marshal the data properly. Consequently, any computer platform and/or language can process the data properly . For example, DDS performs the proper endianess transformations, alignment, and adjustment for 32 versus 64 bit platforms.
Knowing the data type is also required for the proper functioning of **`ContentFilteredTopics`**.

Moreover, explicit definition of the data types is required for the tools provided by DDS vendors to display and manipulate the data properly. Visualization tools, logging and replay, automatic bridging to other middleware, etc., all depend on data type transparency.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Are all the data types the system uses explicitly defined using IDL which is either manually written or generated from equivalent UML or XML representations?

Procedure:

Look for the IDL (or equivalent XML) files used to define the types used by the system.

Example:

```
// File MyTpes.idl
struct MyType
{
   long x;
```

```
    long y;
    string<10> units;
};
```

G1799

Explicitly associate data types to the Data Distribution Service (DDS) Topics within a DDS Domain

Rationale:

A DDS Topic represents a homogeneous collection of data-objects in the **Global Data Space**. All data-objects within a Topic share a common **data-type**. Knowledge of the type associated with the Topic is required for an application to be able to publish and subscribe data on the Topic.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Do all Topics have an explicit association to a data type.

Procedure:

Look for documentation that lists the Topics in use by the system and verify that each Topic has a data type associated with it

Example:

```
<topic>
 <name>Temperature</name>
 <type>TemperatureData</type>
 <description>
    This topic contains a reading of
    a temperature sensor
 </description>
</topic>
<topic>
  . . .
</topic>
```

G1800

Explicitly identify Keys within the Data Distribution Service (DDS) data type that uniquely identify an instance of a data object.

Rationale:

Within each DDS **Domain** (i.e., **Global Data Space**) a data-object is identified by the tuple (**Topic**, Key). The Key is a set of fields within the data type associated with the Topic that the application has tagged to indicate their role in uniquely identifying the data object. For example, if the Topic represents a person to the IRS, the Key may be simply the field containing the social security number.

The proper definition of the key is necessary to allow DDS to implement the **`KEEP_LAST HISTORY`** QoS properly as well as to enforce QoS policies such as **`DEADLINE`**, and **`OWNERSHIP`**. It is also necessary in order for DDS to supply the proper Sample information to the **DataReader**.

All data types require Keys except in the case where the Topic logically represents a single object, for example when the Topic represents a Message Queue.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Does the declaration of the data-type associated with the Topic explicitly designate using one or more of the fields as a Key?

Procedure:

Examine the IDL (or equivalent XML) files used to define the types used by the system to identify the declaration of the data-type associated with each Topic (i.e., see if there are any tags that designate which fields form the Key).

Example:

```
For data types defined using IDL:
struct SensorData
{
```

```
  long    sensor_id; //@key
  float   value;
  string<32> units;
  string<64> location;
};
struct DepartingFlightData
{
  string<8>    airline_code;  //@key
  long         flight_number; //@key
  string<8>    destination_airport_code;
  string<2>    departing_terminal;
  long         departing_gate;
  FlightTime   scheduled_departure_time;
  FlightTime   expected_departure_time;
  string<32>   status;
};
```

G1801

Explicitly define a Topic Quality of Service (QoS) for each Data Distribution Service (DDS) Topic within a DDS Domain.

Rationale:

DDS Topics define the information model of the system. The QoS Policies associated with the Topics define expectations and constraints that all users (**publishers** or **subscribers**) of the Topic should know. Consequently, definition of the Topic QoS is an important part of the system design.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Differentiated Management of Quality-of-Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Messaging within a DDS Domain
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Is there a document that defines the QoS Policies that each Topic uses and does the document that describes the Topics and their associated data types also provide information on the Topic QoS?

Procedure:

Look at the documents that define the Topics in use and their associated data-types and see if they also define the Topic QoS.

Example:

```
Topic: DepartingAircraft
Type: DepartingAircraftStruct
QoS: HISTORY kind=KEEP_LAST
QoS: RELIABILITY kind=RELIABLE
QoS: DEADLINE duration=15minutes
QoS: LIFESPAN duration = 1 hour
Etc.
```

G1802

Catch Data Distribution Service (DDS) events.

Rationale:

DDS uses **listeners** to notify the application of relevant events such as mis-matched Topic definitions, **QoS** violations, lost samples, etc. Normally these events are dispatched to the most specific entity to which they apply (e.g., the affected **DataReader** in the case of the lost sample notification). However under application control the **`DataReader`** can "mask" certain events such that they are propagated to the enclosing container entity (e.g. the **Subscriber** to which the affected **`DataReader`** belongs). The **DomainParticipant** is the ultimate container of all DDS entities and it is therefore important that it handles (e.g., logs) any events that the contained entities have not handled.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe

Evaluation Criteria:

1) Test:

Is a non-nil listener specified when the **`DomainParticipant`** is created?

Procedure:

Look at the arguments passed to the **`create_domain_participant`** operation on the **`DomainParticipantFactory`** and check the values of the listener and mask arguments.

Example:

```
participantFactory
  = TheParticipantFactory;
participant
  = participantFactory->create_participant
      ( QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

G1803

Explicitly define the Data Distribution Service (DDS) Quality of Service (QoS) Policies to describe real-time messaging criteria for Publishers.

Rationale:

DDS relies on the use of a QoS set of characteristics to match publishers with **subscribers**. If the publishers do not specify a QoS policy other than the default, much of the power of DDS publishing is lost and the capabilities of the publisher are not documented.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Differentiated Management of Quality-of-Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Is the **`get_default_publisher_qos`** operation used to create publisher?

Procedure:

Look for the use of the **`get_default_publisher_qos`** operation within the code.

Example:

```
participant
  = participantFactory->create_participant
      ( QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
DDS::PublisherQos publisherQos;
Participant->get_default_publisher_qos
  ( publisherQos );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

2) Test:

Is the **`PUBLISHER_QOS_DEFAULT`** value used to create publishers?

Procedure:

Look for the use of the **PUBLISHER_QOS_DEFAULT** constant within the code.

Example:

```
DDS::Publisher publisher
  = participant->create_publisher
      ( PUBLISHER_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

DDS::STATUS_MASK_ALL is part of DDS 1.3, prior releases require application to use 0x11111111.

G1804

Explicitly define the Data Distribution Service (DDS) Quality of Service (QoS) Policies to describe DataWriter.

Rationale:

DDS relies on the use of QoS characteristics to match a **`DataWriter`** with each **DataReader** of the same **Topic**. If the **`DataWriter`** does not specify a QoS policy other than the default, much of the power of DDS publishing is lost and the capabilities of the **`DataWriter`** are not documented.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Differentiated Management of Quality-of-Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Is the **`get_default_datawriter_qos`** operation used to create participant?

Procedure:

Look for the use of the **`get_default_datawriter_qos`** operation within the code.

Example:

```
DDS::DataWriterQos dataWriterQos;
publisher->get_default_datawriter_qos
  ( dataWriterQos );
DDS::DataWriter dataWriter
  = publisher ->create_datawriter
      ( myTopic,
        dataWriterQos,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

2) Test:

Is the **`DATAWRITER_QOS_DEFAULT`** value used to create **`DataWriter?`**

Procedure:

Look for the use of the **DATAWRITER_QOS_DEFAULT** constant within the code.

Example:

```
DDS::DataWriter dataWriter
  = participant->create_datawriter
      ( myTopic,
        DATAWRITER_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

DDS::STATUS_MASK_ALL is part of DDS 1.3, prior releases require application to use 0x11111111.

G1805

Explicitly define the Data Distribution Service (DDS) Quality of Service (QoS) Policies to describe the behavior of the Subscriber.

Rationale:

DDS relies on the use of QoS set of characteristics to match subscribers with **publishers**. If the subscribers do not specify a QoS policy other than the default, much of the power of DDS subscription and publishing is lost and the requirements of the subscriber are not documented.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Differentiated Management of Quality-of-Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Is the **`SUBSCRIBER_QOS_DEFAULT`** value used to create subscribers?

Procedure:

Look for the use of the **`SUBSCRIBER_QOS_DEFAULT`** constant within the code.

Example:

```
DDS::Publisher publisher
  = participant->create_subscriber
      ( SUBSCRIBER_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

2) Test:

Is the **`get_default_subscriber_qos`** operation used to create subscribers?

Procedure:

Look for the use of the **`get_default_subscriber_qos`** operation within the code.

Example:

```
participant
  = participantFactory->create_participant
      ( QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
DDS::SubscriberQos subscriberQos;
Participant->get_default_subscriber_qos
  ( subscriberQos );
```

DDS::STATUS_MASK_ALL is part of DDS 1.3, prior releases require application to use 0x11111111.

G1806

Explicitly define the Request-Offered Data Distribution Service (DDS) Quality of Service (QoS) Policies to describe the behavior of the DataReader.

Rationale:

DDS relies on the use of QoS characteristics to match a **DataWriter** with each **`DataReader`** of the same Topic. If the **`DataReader`** does not specify a QoS policy other than the default, much of the power of DDS subscription and publishing is lost and the requirements of the **`DataReader`** are not documented.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Differentiated Management of Quality-of-Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Is the **`DATAREADER_QOS_DEFAULT`** value used to create **`DataReader`**?

Procedure:

Look for the use of the **`DATAREADER_QOS_DEFAULT`** constant within the code.

Example:

```
DDS::DataResder dataReader
  = participant->create_datareader
      ( DATAREADER_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

2) Test:

Is the **`get_default_datareader_qos`** operation used to create participant?

Procedure:

Look for the use of the **`get_default_datareader_qos`** operation within the code.

Example:

```
DDS::DataReaderQos dataReaderQos;
publisher->get_default_datareader_qos
  ( dataReaderQos );
DDS::DataReader dataReader
  = publisher ->create_datareader
      ( myTopic,
        dataReaderQos,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

G1807

Check the return values of Data Distribution Service (DDS) functions.

Rationale:

Many of the DDS operations return a nil value when the operation does not work. Not checking for these **`nil`** values can cause unexpected and potentially non-deterministic behavior. Different implementations of the DDS may even behave differently when these values are used. The following is a list of operations that can return **`nil`**:

- create_publisher
- create_subscriber
- create_topic
- create_contentFilteredtopic
- create_multitoic
- find_topic
- lookup_topicdescription
- create_participant
- lookup_participant
- create_datawriter
- lookup_datawriter
- create_datareader
- lookup_datareader
- create_readcondition
- create_querycondition

One operation returns **`HANDLE_NIL`** when the operation fails.

- **`lookup_instance`**

The remaining operations return a **`DDS::ReturnCode_t`** enumerated value that indicates whether the operation succeeded (**`DDS::RETCODE_OK`**) of else the reason for failure.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe

Evaluation Criteria:

1) Test:

Do all invocations of the DDS operations lookup_instance check for a return value of **`HANDLE_NIL`**?

Procedure:

Examine the code for the use of the lookup_instance operations and make sure they check for the return of a **`DDS::HANDLE_NIL`** value immediately after the operation.

Example:

```
DDS::InstanceHandle_t instanceHandle
  = DDS::HANDLE_NIL;
instanceHandle
  = writer->lookup_instance( instance )
if ( instanceHandle == DDS::HANDLE_NIL )
  { cerr << "..."
        << endl;
    exit(1);
  } // End if
```

2) Test:

Are all of the DDS operations that can return **`nil`** values checked for the return of a **`nil`** values?

Procedure:

Examine the code for the use of the following operations and make sure they check for the return of a **`nil`** value immediately after the operation.

- create_publisher
- create_subscriber
- create_topic
- create_contentFilteredtopic
- create_multitoic
- find_topic
- lookup_topicdescription
- create_participant
- lookup_participant
- create_datawriter
- lookup_datawriter
- create_datareader
- lookup_datareader
- create_readcondition
- create_querycondition

Note: *Examine the return of any other operation and make sure they check for* ***`DDS:RETCODE_OK`*** *immediately after the operation.*

Example:

```
DDS::Publisher publisher
  = participant->create_publisher
      ( PUBLISHER_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

```
if ( publisher == NULL ) )
  { cerr << "create_publisher failed."
        << endl;
    exit(1);
  } // End if
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

3) Test:

Are all invocations to DDS operations that return a **`DDS::ReturnCode_t`** checked for **`DDS::RETCODE_OK`**?

Procedure:

Examine the code for the use of the operations with prototype returning **`DDS::ReturnCode_t`** to make sure they check for the return of a **`DDS::RETCODE_OK`** immediately after the operation.

Example:

```
retcode
  = writer->write( … )
if ( retcode != DDS::RETCODE_OK )
  { cerr << "..."
        << endl;
    //  handle error
} // End if
```

G1808

Handle all Data Distribution Service (DDS) Quality of Service (QoS) contract violations using one of the Subscriber access APIs.

Rationale:

QoS contract violations typically indicate either a system mis-configuration, or else a transient failure (e.g., a network that has been temporarily disconnected). Either way the application must monitor these events to determine if they are relevant to their operation and consequently take proper corrective action.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Differentiated Management of Quality-of-Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Are all the DDS QoS-related status change events are captured via a DDS **Listener** or a DDS **WaitSet**?

Procedure:

Specifically ensure that the following DDS events are handled. Look at the arguments passed to the **`create_domain_participant`**, **`create_datawriter`**, and **`create_datareader_operations`** and check that the listener and mask parameters to verify that the following events are being handled:

- OFFERED_DEADLINE_MISSED_STATUS
- REQUESTED_DEADLINE_MISSED_STATUS
- OFFERED_INCOMPATIBLE_QOS_STATUS
- REQUESTED_INCOMPATIBLE_QOS_STATUS
- LIVELINESS_LOST_STATUS
- LIVELINESS_CHANGED_STATUS

Example:

```
participantFactory
  = TheParticipantFactory;
quickQuoterParticipant
  = participantFactory->create_participant
      ( QUICK_QUOTER_DOMAIN_ID,
```

```
          PARTICIPANT_QOS_DEFAULT,
          participantListener,
          DDS::STATUS_MASK_ALL
        );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

G1809

Handle all Data Distribution Service (DDS) events using one of the subscriber access APIs.

Rationale:

Listeners and the dual **Condition**/**WaitSet** infrastructure allow applications to be notified when changes occur in a **DCPS** communication.

Listeners provide a generic mechanism for the middleware to notify the application of relevant asynchronous events, such as arrival of data corresponding to a **subscription**, violation of a **QoS** setting, etc. Each DCPS entity supports its own specialized kind of listener. Listeners are related to changes in status conditions. Listener operations are invoked using a middleware-provided thread.

Conditions and **`WaitSets`** provide the means for an application thread to block waiting for the same events that can be received via a Listener. Using a **`WaitSet`**, the application can handle the event in its own thread instead of the middleware provided thread used for Listeners.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe

Evaluation Criteria:

1) Test:

Are all DDS status change events are captured via a DDS Listener or a DDS WaitSet?

Procedure:

Verify that the following DDS events are handled. Look at the arguments passed to the **`create_domain_participant`**, **`create_datawriter`**, and **`create_datareader_operations`** checking that the listener and mask parameters to verify that the following events are handled:

- **`INCONSISTENT_TOPIC_STATUS`**
- **`SAMPLE_LOST_STATUS`**
- **`SAMPLE_REJECTED_STATUS`**
- **`DATA_ON_READERS_STATUS`**
- **`DATA_AVAILABLE_STATUS`**
- **`OFFERED_DEADLINE_MISSED_STATUS`**
- **`REQUESTED_DEADLINE_MISSED_STATUS`**
- **`OFFERED_INCOMPATIBLE_QOS_STATUS`**

- **REQUESTED_INCOMPATIBLE_QOS_STATUS**
- **LIVELINESS_LOST_STATUS**
- **LIVELINESS_CHANGED_STATUS**

Example:

```
participantFactory
  = TheParticipantFactory;
quickQuoterParticipant
  = participantFactory->create_participant
      ( QUICK_QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        participantListener,
        DDS::STATUS_MASK_ALL
      );
```

DDS::STATUS_MASK_ALL is part of DDS 1.3, prior releases require application to use 0x11111111.

G1810

Use data models to document the data contained within the Data Distribution Service (DDS) Data-Centric Publish Subscribe (DCPS).

Rationale:

DCPS contains static and raw data that can be used is any number of views or objects. As a consequence, changes in the definition of the data, its DDS **Domains** or its structure can have a huge cascading effect. To minimize the impact of these changes, data needs to be documented in a data model that is not subject to implementation.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability

Evaluation Criteria:

1) Test:

Is there a conceptual data model that captures the data within the DCPS?

Procedure:

Determine if there is a conceptual data model that captures the data within the DCPS.

Example:

None.

G1862

Configure Active Directory for Smart Card Logon.

Rationale:

This is a DoD requirement; DoD Instruction 8520.2 [R1206] and DoD Directive 8190.3 [R1297] refer and Joint Task Force-Global Network Operations (JTF-GNO) Communications Tasking Order (CTO 06-02) specifically directs implementation of Smart Card Logon (SCL) on all **NIPRNet** networks.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Smart Card Logon
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Smart Card Logon
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Smart Card Logon
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Smart Card Logon
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Smart Card Logon
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Smart Card Logon

Evaluation Criteria:

1) Test:

Is Active Directory configured for SCL?

Procedure:

Verify that Active Directory is configured for SCL?

Example:

None.

G1869

Configure Domain Controllers for Smart Card Logon.

Rationale:

This is a DoD requirement; DoD Instruction 8520.2 [R1206] and DoD Directive 8190.3 [R1297] refer, and Joint Task Force-Global Network Operations (JTF-GNO) Communications Tasking Order (CTO 06-02) specifically directs implementation of Smart Card Logon (SCL) on all **NIPRNet** networks.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Smart Card Logon
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Smart Card Logon
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Smart Card Logon
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Smart Card Logon
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Smart Card Logon
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Smart Card Logon

Evaluation Criteria:

1) Test:

Is the Domain Controller configured for SCL?

Procedure:

Verify that the Domain Controller is configured for SCL.

Example:

None.

G1883

Use a DoD PKI code signing certificate to sign mobile code residing on DoD-owned or DoD-controlled servers.

Rationale:

DoD Instruction 8552.01 [R1292] requires providing a DoD PKI issued code-signing certificate for all DoD-owned or DoD controlled servers. DoD code-signing certificates must be used to sign mobile code that will reside on DoD servers whenever possible.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Mobile Code

Evaluation Criteria:

1) Test:

Is mobile code residing on a DoD-owned or DoD-controlled server signed by a DoD code signing certificate from an approved DoD PKI Certificate Authority?

Procedure:

Verify that the mobile code has been signed.
Verify that the certificate was issued by a DoD PKI Certificate Authority that issues code signing certificates.

Example:

For signing mobile code using Mozilla/Netscape ***SignTool***:

- How to Sign Applets Using RSA-Signed Certificates: http://java.sun.com/j2se/1.4.2/docs/guide/plugin/developer_guide/rsa_signing.html
- Netscape Certificate Management System Administrator's Guide, Appendix F: http://docs.sun.com/source/816-5531-10/app_sign.htm
- Code Signing Digital IDs for Netscape Object Signing: http://www.verisign.com/resources/gd/objectSigning/index.html

For signing Java applets using Java ***Keytool***:

- How to Sign Applets Using RSA-Signed Certificates: http://java.sun.com/j2se/1.4.2/docs/guide/plugin/developer_guide/rsa_signing.html
- Keytool - Key and Certificate Management Tool: http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html
- Code Signing Digital IDs for Sun Java Signing: http://www.verisign.com/resources/gd/javaSigning/index.html

For signing Microsoft Office ***VBA macros***:

- Code Signing Digital IDs for Microsoft Office 2000/Visual Basic for Applications: http://www.verisign.com/resources/gd/msOffice/index.html

For signing mobile code using Microsoft ***Signcode***:

- Signing and Checking Code With Authenticode: http://msdn.microsoft.com/workshop/security/authcode/signing.asp
- Code Signing Digital IDs for Microsoft Authenticode Technology: http://www.verisign.com/resources/gd/authenticode/index.html

For signing mobile code with Internet Explorer ***Administration Kit 5.0*** or later:

- Code Signing With IEAK 5 and Later: http://support.microsoft.com/default.aspx?scid=kb;en-us;269395

G1884

Configure browsers to use Category 1A allowed mobile code per DoD Instruction 8552.01. [R1292]

Rationale:

DoD Instruction 8552.01 [R1292] requires only allowing ActiveX and Shockwave movies in browsers.

***Note:** Microsoft Internet Explorer version 6/SP2 or version 7 is the only browser that is capable of executing ActiveX controls in compliance with the Category 1 usage restrictions.*

***Note:** The lack of mobile code in a system does not constitute a waiver for the system.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Mobile Code

Evaluation Criteria:

1) Test:

Is the browser properly configured to comply with the Category 1A usage restrictions for ActiveX and Shockwave controls?

Procedure:

Verify configuration of the browser to comply with Category 1A usage restrictions for ActiveX and Shockwave.

Example:

None.

G1885

Configure browsers to disable Category 1X prohibited mobile code per DoD Instruction 8552.01. [R1292]

Rationale:

DoD Instruction 8552.01 [R1292] requires disabling the following prohibited Category 1X mobile code in browsers:

Mobile code scripts that execute in Windows Scripting Host or WSH (e.g., JavaScript and VBScript downloaded via a **Uniform Resource Locator** [**URL**] file reference or email attachment)

- HTML Applications (e.g., `.HTA` files) that download as mobile code
- Scrap objects
- Microsoft Disk Operating System (MS-DOS) batch scripts
- Unix shell scripts
- Binary executables (e.g., `.exe` files) that download as mobile code

> ***Note:*** *The lack of mobile code in a system does not constitute a waiver for the system.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Mobile Code

Evaluation Criteria:

1) Test:

Is the browser properly configured to disable Category 1X prohibited mobile code?

Procedure:

Verify all Category 1X prohibited mobile code is disabled in the browser.

Example:

None.

G1886

Disable automatic execution of mobile code in email clients.

Rationale:

Due to the significant risk of malicious mobile code downloading into user workstations via email, and the ease of rapidly spreading malicious mobile code via email, the following restrictions apply to all types of mobile code in email independent of risk category:

- Disable the automatic execution of all categories of mobile code in email bodies and attachments .
- Configure desktop software to prompt the user prior to opening email attachments that may contain mobile code.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Mobile Code

Evaluation Criteria:

1) Test:

Is automatic execution of mobile code in email bodies and attachments disabled?

Procedure:

Verify that Category 1X mobile code file types have been disassociated.
Verify that execution of mobile code is disabled in an email body

Verify that execution of mobile code is disabled in an email attachment.

Example:

Some email client products, such as Microsoft Outlook and Outlook Express, use the Windows file type associations to select the appropriate application to process a file. Disassociating these file types in Windows will prevent the contents of files with those related file extensions from automatically executing whenever the user selects the file.

2) Test:

Is the user prompted prior to opening email attachments?

Procedure:

Verify that the user is prompted prior to opening email attachments containing mobile code.

Example:

DoD mobile code policy requires prompting the user prior to opening email attachments that may contain mobile code. Microsoft Outlook Express and Outlook use the Windows file types and settings. SeaMonkey and Thunderbird maintain their own internal file type settings. Windows should be configured to prompt users prior to opening downloaded files. In addition, Windows must be configured to always display all files and file extensions to enable users to determine the type of file they may be opening.

G1887

Monitor configured mobile code-enabled software to ensure it is in compliance with DoD Instruction 8552.01. **[R1292]**

Rationale:

The primary foundation for implementing the DoD Mobile Code Policy and protecting against malicious mobile code is the proper secure configuration of users' desktop workstation software. The policy requires immediate correction of all identified misconfigurations.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Mobile Code

Evaluation Criteria:

1) Test:

Is there a plan or process in place to configure mobile code properly on DoD systems?

Procedure:

Verify configuration of workstation and server mobile code-enabled software to be compliant with DoD Instruction 8552.01. [R1292]
Verify that all identified misconfigurations are corrected immediately.

Example:

None.

G1895

Encrypt all Unclassified DoD Data at Rest (DAR) not releasable to the public stored on mobile computing devices.

Rationale:

DoD mandates encryption not only for Personally Identifiable Information (PII), but for all non-publicly released Unclassified information that is contained on mobile computing devices and removable media.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Data at Rest

Evaluation Criteria:

1) Test:

Is all non-publicly released Unclassified information contained on mobile computing devices and removable storage media encrypted?

Procedure:

Verify that a data at rest encryption product is properly installed and configured to encrypt DAR on mobile computing devices and removable storage media containing non-publicly released Unclassified information.

Example:

None.

G1896

Use Data at Rest (DAR) products that are Federal Information Processing Standard (FIPS) 140-2 compliant.

Rationale:

The Office of Management and Budget (OMB) and **DoD** require that all encryption products meet **National Institute of Standards and Technology** (**NIST**) **Federal Information Processing Standard** (**FIPS**) 140-2 requirements or have a **National Security Agency** (**NSA**) approval letter for use in U.S. Government networks.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Data at Rest

Evaluation Criteria:

1) Test:

Is the DAR encryption FIPS 140-2 compliant?

Procedure:

Verify that NIST has validated the cryptographic module to meet NIST FIPS 140-2 requirements or NSA has approved the module for use on government networks.

Example:

None.

G1897

Purchase Data at Rest (DAR) encryption products that are included in the Enterprise Software Initiative (ESI).

Rationale:

DoD components must purchase **data at rest** (**DAR**) encryption products to protect DAR on mobile computing devices and removable storage media through the Enterprise Software Initiative (ESI) since it benefits all of the DoD. All ESI awarded products are **Federal Information Processing Standard** (**FIPS**) 140-2 compliant, support **Common Access Card** (**CAC**) integration, licenses are transferable within a federal agency, and licenses include secondary use rights.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Data at Rest

Evaluation Criteria:

1) Test:

Are DAR encryption products purchased through the ESI?

Procedure:

Verify that DAR encryption products are purchased through the ESI.

Example:

None.

G1902

Use the Exclusive Canonicalization algorithm when digitally signing XML content that may be embedded in another XML document.

Rationale:

Namespaces are inherited from parent XML nodes. The digital signature of a signed XML message fragment from a source XML document placed into destination XML document may fail signature verification due to the inheritance of namespaces from the destination document. Exclusive canonicalization handles namespaces of surrounding XML content differently to support this use case.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures

Evaluation Criteria:

1) Test:

Are XML message fragments intented for inclusion in other XML documents canonicalized using the Exclusive Canonicalization algorithm?

Procedure:

Verify message fragments intented for inclusion in other XML documents are canonicalized using the Exclusive Canonicalization algorithm by inspecting the canonicalization method in the source code or the resulting signed XML **`CanonicalizationMethod`** element.

Example:

None.

G1910

Provide for transformation of XML messages using eXtensible Style Language Transformations (XSLT) when implementing an Enterprise Service Bus (ESB).

Rationale:

Mediation, including transformation, is a core characteristic of an ESB. XSLT is the most commonly used standards-based language for transforming XML.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Enterprise Service Bus (ESB)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Enterprise Service Bus (ESB)
NESI / Part 5: Developer Guidance / Middleware / Enterprise Service Bus (ESB)

Evaluation Criteria:

1) Test:

Does the implemented ESB architecture provide for transformation of XML messages using XSLT?

Procedure:

Verify that the implemented ESB architecture provide for transformation of XML messages using XSLT?

Example:

None.

G1912

Support the execution of a formally specified Business Process Execution Language (BPEL) when implementing an Enterprise Service Bus (ESB).

Rationale:

A BPEL, such as WS-BPEL [R1347], is an orchestration language for executing business process through the arrangement, coordination and management of services into composite services. BPEL orchestration capabilities within an ESB allows for a standards-based way to execute buisness processes. Business process orchestration outside of an ESB may lead to duplicative and conflicting mediation and registration capabilities.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Enterprise Service Bus (ESB)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Enterprise Service Bus (ESB)
NESI / Part 5: Developer Guidance / Middleware / Enterprise Service Bus (ESB)

Evaluation Criteria:

1) Test:

Does the ESB provide support for the execution of BPEL?

Procedure:

Verify that the ESB is able to execute BPEL.

Example:

None.

G1942

Provide applications the ability to export Public Key Infrastructure (PKI) software certificates.

Rationale:

The whole **Public Key Infrastructure** (**PKI**) system is predicated on the use of public-private key pairs. The ability to import (recover) and export (backup) key pairs is critical to a functional PKI application.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.5, Version 1.0, 13 July 2000.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / Key Management
NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test:

Is the application able to export its key pair for backup/recovery purposes?

Procedure:

Have the application export a key pair.

> ***Note:*** *Verify the correctness of the exported file through analysis.*

Example:

Internet Explorer can import/export certificates using Tools > Internet Options. Click on Internet tab and then click on Certificates link. Import/Export options are located here.

UNIX-based Web server keys are exported by making a copy of the keys file and placing it in a safe location.

BP1007

Develop software using open standard Application Programming Interfaces (APIs).

Rationale:

Using open standard APIs enables code portability and reduces dependancies on proprietary APIs.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Does the application create customized/proprietary solutions where standardized **API**s exists?

Procedure:

Check the application for code that has proprietary solutions where standardized APIs exists.

Example:

None

BP1021

Create fully encapsulated classes.

Rationale:

Data members should not be public as making implementation details public creates interdependencies between the class and its users, subjecting the users to changes in implementation. Therefore, access should only occur via public interface methods. This makes the implementation more robust, because all data can be validated when assigned new values and allows for logging of changed values.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Maintainability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test:

Do instance variables have public access or are they more accessible than necessary?

Procedure:

Check that the instance variable in classes does not have public access unless it is static and final.

Example:

None

2) Test:

Does the class provide direct access to internal data via pass by reference?

Procedure:

Check to make sure that the methods that access the internal state do not return a reference to the internal data.

Example:

None

BP1038

Use a sans serif font (e.g., Arial, Verdana) in Web pages rather than a serif font (e.g., Times New Roman).

Rationale:

Web pages are easier to read with **sans serif** fonts.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces

BP1039

Do not underline any text unless it is a link.

Rationale:

Underlined text is the default behavior of an **HTML** link. Many users consider this the norm and may find a **Web page** difficult to read if other items are underlined.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces

BP1040

Use hex codes for all colors (e.g., #FFFF33), never the color name (e.g., yellow).

Rationale:

Using hex codes for colors is a common industry practice to increase compatibility between browsers.

For an online hexadecimal color chart, see http://webmonkey.wired.com/webmonkey/reference/color_codes/.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients

BP1041

Do not change the default colors of the links.

Rationale:

Web pages are easier to read because users have become accustomed to the default colors.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Style Sheets
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces

BP1042

Do not build a Web page where the horizontal width is greater than the screen (vertical scrolling is fine), planning for the lowest common denominator to be super-VGA resolution (800 x 600).

Rationale:

This enables a user to print pages on most printers and render pages on most displays.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces

BP1054

Use conventional user interface controls that provide input choices for the user.

Rationale:

Using conventional controls such as radio buttons, check boxes, list boxes, and drop-downs reduces user input errors and aids in data integrity.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

BP1097

Use the `System.Text.StringBuilder` class for repetitive string modifications such as appending, removing, replacing, or inserting characters.

Rationale:

Strings in **.NET** are immutable. This means that every time a string is created as a result of a string operation such as concatenation, a new string is created for each intermediate string in a set of operations. This has a lot of string management overhead. **`StringBuilder`** avoids these problems.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / .NET Framework
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / .NET Framework
NESI / Part 5: Developer Guidance / Middleware / .NET Framework

Evaluation Criteria:

1) Test:

Are there repetitive string operations that use string operations instead of **`StringBuilder`** operations?

Procedure:

Scan all C# code for repetitive string operations such as appending, removing, replacing, or inserting characters.

Example:

None

BP1098

Write all .NET code in C#.

Rationale:

Because of the high degree of similarities between C# and Java, .NET code written in C# is easily ported to Java. .NET has removed most of the advantages of one language (C#, C++, J++, VB) over another.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / .NET Framework
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / .NET Framework
NESI / Part 5: Developer Guidance / Middleware / .NET Framework

Evaluation Criteria:

1) Test:

Are any .NET languages delivered other than C#?

Procedure:

Scan delivered code for registered .NET file extensions other than C#.

Example:

None

BP1100

Compile all .NET code using the .NET Just-In-Time compiler.

Rationale:

There are two different ways to generate machine code within the .NET environment: **Just-In-Time (JIT)** and **Native Image Generator** (**NGEN**). The NGEN method provides performance advantages by using the native image cache portion of the global assembly cache, which is specific to the machine where the .NET **common language runtime** is installed. It is machine-dependent and is less portable.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / .NET Framework
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / .NET Framework
NESI / Part 5: Developer Guidance / Middleware / .NET Framework

Evaluation Criteria:

1) Test:

Is **`ngen.exe`** used?

Procedure:

Scan all delivered code for the use of **`ngen.exe`** or the **ngen** command.

Example:

None

BP1111

Mark all Microsoft Message Queue (MSMQ) messages as recoverable.

Rationale:

MSMQ normally only stores the contents of messages in memory, which will be lost if a power, hardware, or software failure occurs. By marking messages as recoverable, messages are also stored to disk so the contents can be recovered after a failure.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Messaging with MSMQ
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Messaging with MSMQ
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Messaging with MSMQ
NESI / Part 5: Developer Guidance / Middleware / Messaging / Messaging with MSMQ

Evaluation Criteria:

1) Test:

Are all messages and message queues marked as recoverable?

Procedure:

Scan the code for the creation of messages and message codes, and make sure each has the `recoverable` attribute set to true.

Example:

None

BP1112

Specify all Microsoft Message Queue (MSMQ) queues as transactional if they support multiple-step processes.

Rationale:

Transactions allow multi-step processes to behave correctly when a **rollback** occurs.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Messaging with MSMQ
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Messaging with MSMQ
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Messaging with MSMQ
NESI / Part 5: Developer Guidance / Middleware / Messaging / Messaging with MSMQ

BP1116

If using Java-based messaging (e.g., JMS), register destinations in Java Naming and Directory Interface (JNDI) so message clients can use JNDI to look up these destinations.

Rationale:

JNDI is an industry standard for Java-based applications. Many JMS interoperability coding issues relate to the publication and discovery of JNDI for resources. To mitigate these issues, encapsulate resource definitions in a properties file or in **Java EE** as a **deployment descriptor**.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / JNDI Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / JNDI Security

BP1139

Do not use proprietary SQL extensions.

Rationale:

The use of proprietary extensions increases vendor dependence.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

Evaluation Criteria:

1) Test:

Have the developers adhered to a core set of features and minimized use of proprietary extensions to the **SQL** standard?

Procedure:

Examine a representative sample of database scripts and stored procedures.

Example:

None

BP1140

Use SQL-2003 features in preference to SQL-92 or SQL-99.

Rationale:

SQL-2003 includes many **XML** and **OODB** extensions and features. Use it in preference to SQL-99 or SQL-92 entry-level features to justify the recommendations against using native XML databases and OODB databases.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

Evaluation Criteria:

1) Test:

Have the developers used SQL-2003 features rather than SQL-92 or SQL-99 features?

Procedure:

Examine a representative sample of database scripts and stored procedures.

Example:

None

BP1143

Use a database modeling tool that supports a two-level model (Conceptual/Logical and Physical) and ISO-11179 data exchange standards.

Rationale:

ISO-11179 is a **metadata** repository standard. Supporting tools store the model locally in an **XML** file or in a vendor-specific repository. For many applications, there is no need to use the repository at all. **Configuration Management** could be affected by checking the model in and out of a tool such as Source Safe. Entity-Relationship **data model** is synonymous with a **Conceptual data model**.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

Evaluation Criteria:

1) Test:

Is a database modeling tool being used and does it support the **ISO-11179** data exchange standards?

Procedure:

Verify that the requirement for a database modeling tool is included in the system requirements. If ISO-11179 standard-based repository products become available, determine whether the product provides an interface thereto.

Example:

None

BP1145

Use vendor-neutral conceptual/logical models.

Rationale:

The leading database vendors do not have a common set of data types or object name length limitations, and there are no ANSI standards that address these issues. To maintain vendor-neutral models, do not accept vendor-specific features.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling

Evaluation Criteria:

1) Test:

Has the data model been designed using vendor-neutral design criteria?

Procedure:

Examine the conceptual/logical data model.

Example:

None

BP1227

Do not allow installation of MSMQ-dependent clients.

Rationale:

MSMQ-dependent clients require synchronous access to an MSMQ server and create performance issues on the server. Consequently, dependent clients cannot operate if they are disconnected from the rest of the **enterprise** networks.

Dependent clients cannot be run under local accounts.

Dependent clients leave all encrypted messages in plain text between the client and server.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Messaging with MSMQ
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Messaging with MSMQ
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Messaging with MSMQ
NESI / Part 5: Developer Guidance / Middleware / Messaging / Messaging with MSMQ
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

BP1230

Do not use the MSMQ `SupportLocalAccountsOrNT4` feature.

Rationale:

This entry enables weakened security for Active Directory on a **domain** controller, which is then replicated to all other domain controllers in every domain in your forest.

See the Microsoft Message Queuing Web site for additional information.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Messaging with MSMQ
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Messaging with MSMQ
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Messaging with MSMQ
NESI / Part 5: Developer Guidance / Middleware / Messaging / Messaging with MSMQ

BP1231

Use `CORBA::String_var` in IDL to pass string types in C++.

Rationale:

Follow this practice to correct memory management and reduce memory leaks and runtime faults.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA

Evaluation Criteria:

1) Test:

Is **`String_var`** used in the implementation code that was not auto generated?

Procedure:

Check implementation code that was not autogenerated for all occurrences of "string" and verify that they are **`String_var`** .

Example:

None

BP1232

Do not pass or return a zero or null pointer; instead, pass an empty string.

Rationale:

Follow this practice to correct memory management and reduce memory leaks and runtime faults.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA

Evaluation Criteria:

1) Test:

Are there any returns that contain pointers that are assigned zero?

Procedure:

Check code to make sure that all strings returned always have a safety check for zero or null pointers, and assign them to empty strings.

Example:

None

BP1233

Do not assign `CORBA::String_var` type to `INOUT` method parameters.

Rationale:

Follow this practice to correct memory management and reduce memory leaks and runtime faults.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA

Evaluation Criteria:

1) Test:

Are there any implementation classes using methods that contain **`CORBA::String_var`**?

Procedure:

Inspect CORBA code to make sure **`INOUT`** parameters are not assigned to **`CORBA::String_var`** values.

Example:

None

BP1234

Assign string values to `OUT` , `INOUT` , or `RETURN` parameters using operations to allocate or duplicate values rather than creating and deleting values.

Rationale:

Correct memory management and reduce memory leaks and reduce runtime faults.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA

Evaluation Criteria:

1) Test:

Are **`string_dup`**, **`string_alloc`** and **`string_free`** being used?

Procedure:

Search CORBA code for the use of **`string_dup`**, **`string_alloc`**, and **`string_free`**.

Example:

None

2) Test:

Are new and delete operators being used for strings being assigned to **`OUT`**, **`INOUT`**, or **`RETURN`** parameters?

Procedure:

Inspect CORBA code to make sure **`OUT`**, **`INOUT`**, and **`RETURN`** parameters are not using strings managed with the new and delete operators.

Example:

None

BP1235

Assign string values to returned-as-attribute values using operations to allocate or duplicate values rather than creating and deleting values.

Rationale:

Follow this practice to correct memory management and reduce memory leaks and runtime faults.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / CORBA
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / CORBA
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / CORBA
NESI / Part 5: Developer Guidance / Middleware / CORBA

Evaluation Criteria:

1) Test:

Are **`string_dup`**, **`string_alloc`**, and **`string_free`** being used?

Procedure:

Search CORBA code for the use of **`string_dup`**, **`string_alloc`**, and **`string_free`**.

Example:

None

2) Test:

Are new and delete operators being used for strings being returned-as-attribute?

Procedure:

Inspect CORBA code to make sure returned-as-attribute string values are not using strings managed with the new and delete operators.

Example:

None

BP1240

Present complete and coherent sets of concepts to the user.

Rationale:

The **interface** should not require the consumer continually to implement multiple interfaces when a single interface can accomplish the same thing.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design

BP1241

Design statically typed interfaces.

Rationale:

Designing a statically typed interface allows consumers to use early binding rather than late binding. This minimizes the risk for runtime errors due to late binding.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design

BP1242

Minimize an interface's dependencies on other interfaces.

Rationale:

Minimizing the dependency of an interface on other interfaces simplifies the use of the interface by consumers.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design

BP1243

Express interfaces in terms of application-level types.

Rationale:

Use application-level types to maintain the meaning of values used with the interface. This enables data validation and other runtime safety checks against the data.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / C4ISR: Payload Platform / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Public Interface Design
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Public Interface Design
NESI / Part 5: Developer Guidance / Public Interface Design

BP1246

Base Java-based portlets on JSR 168.

Rationale:

JSR 168 enables **interoperability** between Java **portlets** and **portals**. This specification defines a set of **APIs** for portal computing that addresses the areas of aggregation, **personalization**, presentation, and security. http://www.jcp.org/en/jsr/detail?id=168

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Web Portals
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Web Portals
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Web Portals

BP1247

Encapsulate Java-based portlets in a Web Application Archive (.war) file.

Rationale:

Storing JSR-168-compliant code in the portal container improves **interoperability** and code reuse.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients / Web Portals
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients / Web Portals
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients / Web Portals

BP1248

Follow a naming convention.

Rationale:

The names of schemas, users, tables, and columns need to be unique and descriptive. Unfortunately, it is possible (but undesirable) to give the same name to multiple objects; for example, assigning the name "employee" to a database, table, and column. Many naming conventions get around this by appending a suffix that indicates the kind of object: for example, **`Employee_Db`**, **`Employee_Tbl`**, **`Employee_Id`**, **`Employee_Indx`**.

Avoid generic column names such as "ID." Systems often have many kinds of IDs, and even if the system really only does have a single ID, it will be more difficult to merge with other databases if they have also used the column name "ID."

Some DBMSs support mixed-case names of unlimited length, while others are case-insensitive. For portability, assume that names are case-insensitive and limited to 30 characters. Do not use reserved words from the **SQL-92**, **SQL:1999**, or SQL:2003 standards.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

Evaluation Criteria:

1) Test:

Is there a naming convention?

Procedure:

Check for the existence of a document that governs naming conventions, or look for patterns in the database metadata.

Example:

Use database commands to look at the database metadata:

```
select username from all_users
select table_name from user_tables
select index_name from user_indexes
```

BP1249

Do not use generic names for database objects such as databases, schema, users, tables, views, or indices.

Rationale:

Assigning generic names to user-defined objects within a database can lead to confusion and unexpected results. For example, naming a database "instance" within the **RDBMS** database is confusing to the humans who have to read commands that reference the database. In addition, the RDBMS software may parse it incorrectly.

> ***Note:*** *Although some RDBMS interpreters allow the use of a generic or reserved word to name objects if the name is surrounded with quotes, this is not a recommended practice.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

Evaluation Criteria:

1) Test:

Are any generic names used for user-defined objects?

Procedure:

Examine the RDBMS metadata for generic names such as database, table, entity, column, attribute, select, view, etc.

Example:

```
select table_name from user_tables where table_name in ('database','entity',...)
select column_name from user_tab_columns where column_name in ('database','entity',...)
```

BP1250

Use case-insensitive names for database objects such as databases, schema, users, tables, views, and indices.

Rationale:

The **SQL** standard does not require names to be case-sensitive. Consequently, some DBMSs are not case-sensitive. Using case-sensitive names, therefore, makes portability more difficult.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

Evaluation Criteria:

1) Test:

Are the names of database objects case-sensitive?

Procedure:

Examine the database metadata for "run-on" names. If the database supports case-sensitive names, check to see if it is using camel-back capitalization.

Example:

```
EMPLOYEEBENEFITSTBL
EmployeeBenefitsTbl
```

BP1251

Separate words with underscores.

Rationale:

The **SQL** standard does not require names to be case-sensitive. Consequently, some DBMSs are not case-sensitive. Using case-sensitive names, therefore, makes portability more difficult. To avoid these problems, use underscores to separate words (**`employee_benefits_tbl`**) rather than camel-back capitalization (**`EmployeeBenefitsTbl`**).

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

Evaluation Criteria:

1) Test:

Are underscores used between the words in the names of database objects?

Procedure:

Examine the database metadata and look for names that do not have underscores separating words.

Example:

```
EMPLOYEEBENEFITSTBL versus
EMPLOYEE_BENEFITS_TBL
EmployeeBenefitsTbl versus
Employee_Benefits_Tbl
```

BP1252

Do not use names with more than 30 characters.

Rationale:

Not all DBMSs support unlimited name lengths. For example, Oracle limits object names to 30 characters. Therefore, using names longer than 30 characters can reduce portability by limiting the DBMSs on which the system can be deployed.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

Evaluation Criteria:

1) Test:

Are any of the database object names more than 30 characters in length?

Procedure:

Examine the database metadata and look for names that are longer than 30 characters.

Example:

```
W2_EMPLOYEE_BENEFITS_FOR_FAMILIES_TBL
```

BP1253

Do not use the SQL:1999 or SQL:2003 reserved words as names for database objects such as databases, schema, users, tables, views, or indices.

Rationale:

Using reserved words as the names of database objects can cause ambiguities and errors. It limits the ability to upgrade or port the code to other systems.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

Evaluation Criteria:

1) Test:

Are any of the **SQL:1999** or SQL:2003 reserved words used to name objects in the database?

Procedure:

Examine the database metadata for names that are in the list of SQL:1999 or SQL:2003 reserved words

Example:

Look for any of these words:

```
ABS ABSOLUTE ACCESS ACQUIRE ACTION ADA ADD ADMIN AFTER AGGREGATE ALIAS ALL ALLOCATE ALLOW ALTER AND ANY ARE
ARRAY AS ASC ASENSITIVE ASSERTION ASUTIME ASYMMETRIC AT ATOMIC AUDIT AUTHORIZATION AUX AUXILIARY AVG
BACKUP BEFORE BEGIN BETWEEN BIGINT BINARY BIT BIT_LENGTH BLOB BOOLEAN BOTH BREADTH BREAK BROWSE BUFFERPOOL
BULK BY
CALL CALLED CAPTURE CARDINALITY CASCADE CASCADED CASE CAST CATALOG CCSID CEIL CEILING CHAR CHAR_LENGTH
CHARACTER CHARACTER_LENGTH CHECK CHECKPOINT CLASS CLOB CLOSE CLUSTER CLUSTERED COALESCE COLLATE COLLATION
COLLECT COLLECTION COLLID COLUMN COMMENT COMMIT COMPLETION COMPRESS COMPUTE CONCAT CONDITION CONNECT
CONNECTION CONSTRAINT CONSTRAINTS CONSTRUCTOR CONTAINS CONTAINSTABLE CONTINUE CONVERT CORR CORRESPONDING
COUNT COUNT_BIG COVAR_POP COVAR_SAMP CREATE CROSS CUBE CUME_DIST CURRENT CURRENT_COLLATION CURRENT_DATE
CURRENT_DEFAULT_TRANSFORM_GROUP CURRENT_LC_PATH CURRENT_PATH CURRENT_ROLE CURRENT_SERVER CURRENT_TIME
CURRENT_TIMESTAMP CURRENT_TIMEZONE CURRENT_TRANSFORM_GROUP_FOR_TYPE CURRENT_USER CURSOR CYCLE
DATA DATABASE DATALINK DATE DAY DAYS DB2GENERAL DB2SQL DBA DBCC DBINFO DBSPACE DEALLOCATE DEC DECIMAL DECLARE
DEFAULT DEFERRABLE DEFERRED DELETE DENSE_RANK DENY DEPTH DEREF DESC DESCRIBE DESCRIPTOR DESTROY DESTRUCTOR
DETERMINISTIC DIAGNOSTICS DICTIONARY DISALLOW DISCONNECT DISK DISTINCT DISTRIBUTED DLNEWCOPY DLPREVIOUSCOPY
DLURLCOMPLETE DLURLCOMPLETEONLY DLURLCOMPLETEWRITE DLURLPATH DLURLPATHONLY DLURLPATHWRITE DLURLSCHEME
DLURLSERVER DLVALUE DO DOMAIN DOUBLE DROP DSSIZE DUMMY DUMP DYNAMIC
EACH EDITPROC ELEMENT ELSE ELSEIF END END-EXEC EQUALS ERASE ERRLVL ESCAPE EVERY EXCEPT EXCEPTION EXCLUSIVE
EXEC EXECUTE EXISTS EXIT EXP EXPLAIN EXTERNAL EXTRACT
FALSE FENCED FETCH FIELDPROC FILE FILLFACTOR FILTER FINAL FIRST FLOAT FLOOR FOR FOREIGN FORTRAN FOUND FREE
FREETEXT FREETEXTTABLE FROM FULL FUNCTION FUSION
GENERAL GENERATED GET GLOBAL GO GOTO GRANT GRAPHIC GROUP GROUPING
HANDLER HAVING HOLD HOLDLOCK HOST HOUR HOURS
IDENTIFIED IDENTITY IDENTITY_INSERT IDENTITYCOL IF IGNORE IMMEDIATE IMPORT IN INCLUDE INCREMENT INDEX
INDICATOR INITIAL INITIALIZE INITIALLY INNER INOUT INPUT INSENSITIVE INSERT INT INTEGER INTEGRITY INTERSECT
INTERSECTION INTERVAL INTO IS ISOBID ISOLATION ITERATE
JAR JAVA JOIN
KEY KILL
LABEL LANGUAGE LARGE LAST LATERAL LC_CTYPE LEADING LEAVE LEFT LESS LEVEL LIKE LIMIT LINENO LINKTYPE LN LOAD
LOCAL LOCALE LOCALTIME LOCALTIMESTAMP LOCATOR LOCATORS LOCK LOCKSIZE LONG LOOP LOWER
```

```
MAP MATCH MAX MAXEXTENTS MEMBER MERGE METHOD MICROSECOND MICROSECONDS MIN MINUS MINUTE MINUTES MOD MODE
MODIFIES MODIFY MODULE MONTH MONTHS MULTISET
NAME NAMED NAMES NATIONAL NATURAL NCHAR NCLOB NEW NEXT NHEADER NO NOAUDIT NOCHECK NOCOMPRESS NODENAME
NODENUMBER NONCLUSTERED NONE NORMALIZE NOT NOWAIT NULL NULLIF NULLS NUMBER NUMERIC NUMPARTS
OBID OBJECT OCTET_LENGTH OF OFF OFFLINE OFFSETS OLD ON ONLINE ONLY OPEN OPENDATASOURCE OPENQUERY OPENROWSET
OPENXML OPERATION OPTIMIZATION OPTIMIZE OPTION OR ORDER ORDINARILITY OUT OUTER OUTPUT OVER OVERLAPS OVERLAY
PACKAGE PAD PAGE PAGES PARAMETER PARAMETERS PART PARTIAL PARTITION PASCAL PATH PCTFREE PCTINDEX PERCENT
PERCENT_RANK PERCENTILE_CONT PERCENTILE_DISC PIECESIZE PLAN POSITION POSTFIX POWER PRECISION PREFIX PREORDER
PREPARE PRESERVE PRIMARY PRINT PRIOR PRIQTY PRIVATE PRIVILEGES PROC PROCEDURE PROGRAM PSID PUBLIC
QUERYNO
RAISERROR RANGE RANK RAW READ READS READTEXT REAL RECONFIGURE RECOVERY RECURSIVE REF REFERENCES REFERENCING
REGR_AVGX REGR_AVGY REGR_COUNT REGR_INTERCEPT REGR_R2 REGR_SLOPE REGR_SXX REGR_SXY REGR_SYY RELATIVE RELEASE
RENAME REPEAT REPLICATION RESET RESIGNAL RESOURCE RESTORE RESTRICT RESULT RETURN RETURNS REVOKE RIGHT ROLE
ROLLBACK ROLLUP ROUTINE ROW ROW_NUMBER ROWCOUNT ROWGUIDCOL ROWID ROWNUM ROWS RRN RULE RUN
SAVE SAVEPOINT SCHEDULE SCHEMA SCOPE SCRATCHPAD SCROLL SEARCH SECOND SECONDS SECQTY SECTION SECURITY SELECT
SENSITIVE SEQUENCE SESSION SESSION_USER SET SETS SETUSER SHARE SHUTDOWN SIGNAL SIMILAR SIMPLE SIZE SMALLINT
SOME SOURCE SPACE SPECIFIC SPECIFICTYPE SQL SQLCA SQLCODE SQLERROR SQLEXCEPTION SQLSTATE SQLWARNING SQRT
STANDARD START STATE STATEMENT STATIC STATISTICS STAY STDDEV_POP STDDEV_SAMP STOGROUP STORES STORPOOL
STRUCTURE STYLESUBPAGES SUBSTRING SUCCESSFUL SUM SYMMETRIC SYNONYM SYSDATE SYSTEM SYSTEM_USER
TABLE TABLESPACE TEMPORARY TERMINATE TEXTSIZE THAN THEN TIME TIMESTAMP TIMEZONE_HOUR TIMEZONE_MINUTE TO TOP
TRAILING TRAN TRANSACTION TRANSLATE TRANSLATION TREAT TRIGGER TRIM TRUE TRUNCATE TSEQUAL TYPE
UID UNDER UNDO UNION UNIQUE UNKNOWN UNNEST UNTIL UPDATE UPDATETEXT UPPER USAGE USE USER USING
VALIDATE VALIDPROC VALUE VALUES VAR_POP VAR_SAMP VARCHAR VARCHAR2 VARIABLE VARIANT VARYING VCAT VIEW VOLUMES
WAITFOR WHEN WHENEVER WHERE WHILE WIDTH_BUCKET WINDOW WITH WITHIN WITHOUT WLM WORK WRITE WRITETEXT
YEAR YEARS
ZONE
```

BP1254

For command-and-control systems, use the names defined in the Joint Command, Control and Consultation Information Exchange Data Model (JC3IEDM) for data exposed to the outside communities.

Rationale:

The **Command-and-Control** (**C2**) **COI** has developed a **data model** to facilitate the exchange of data within the community and by consumers of their data outside the community. Therefore, data that is to be exposed from the database to the COI community or its data consumers should defer to the **data model** whenever possible. The JC3IEDM [R1070] data model defines the data units as well as the names and structure of the data.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling

Evaluation Criteria:

1) Test:

If this is a system, does it use for the data that is exposed to the outside world?

Procedure:

Review all the data that is exposed to the outside world and confirm that it conforms to the JC3IEDM specifications.

Example:

None

BP1255

Use surrogate keys.

Rationale:

A surrogate key, also referred to as a system-generated key, database-sequence number, or arbitrary unique identifier, is a unique, arbitrary **primary key**. The **RDBMS** usually generates the surrogate key, but a database access layer such as the middle tier can also generate the surrogate key. The surrogate key is arbitrary because it is not derived from any data that exists within the table or the database. Another option for surrogate keys is Universally Unique Identifiers (UUIDs) (http://en.wikipedia.org/wiki/Universally_Unique_Identifier), the most common implementation being Microsoft's Globally Unique Identifiers (GUIDs) (http://en.wikipedia.org/wiki/Globally_Unique_Identifier).

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

BP1256

Use surrogate keys as the primary key.

Rationale:

Instead of using the natural keys to identify each record uniquely, use a surrogate key. This allows the natural key information to be modified independently of the primary key and any foreign-key references to the key.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

Evaluation Criteria:

1) Test:

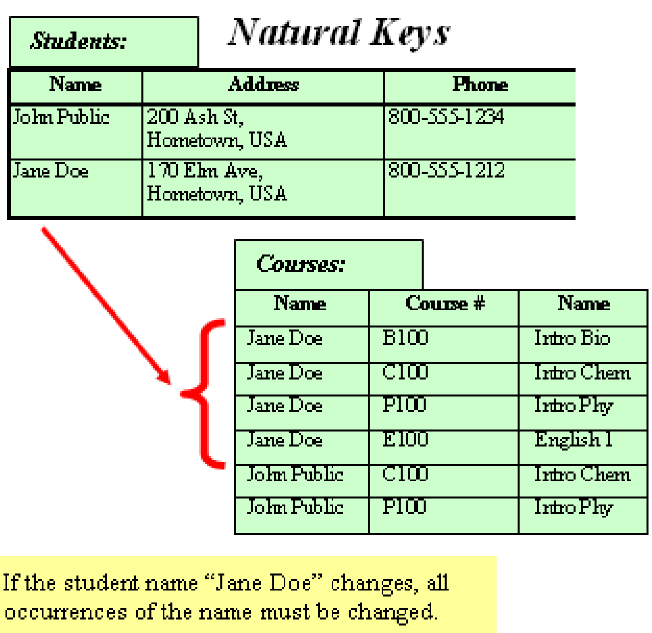
Are surrogate keys used instead of natural keys?

Procedure:

Look at the database metadata and determine if it uses surrogate or natural keys.

Example:

The following example shows natural keys. The primary keys are made up completely or in part from naturally occurring data in the tables.



I1120

The following example shows a surrogate key being used instead of a natural key. Maintaining data is less complex than it is with natural keys and consequently less error-prone.

Surrogate Keys

Students:

Stu. ID	Name	Address	Phone
4321	John Public	200 Ash St, Hometown, USA	800-555-1234
1234	Jane Doe	170 Elm Ave, Hometown, USA	800-555-1212

Courses:

Stu. ID	Course #	Name
1234	B100	Intro Bio
1234	C100	Intro Chem
1234	P100	Intro Phy
1234	E100	English I
4321	C100	Intro Chem
4321	P100	Intro Phy

If the student name “Jane Doe” changes, only one occurrence of the name must be changed.

I1121

BP1257

Place a unique key constraint on the natural key fields.

Rationale:

Surrogate keys make it easier to maintain data. However, a column or set of columns should still uniquely identify the row in the table. This column or set of columns is the "natural key" or "secondary key." This natural key should still be protected by the uniqueness constraint normally associated with a **primary key**.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

Evaluation Criteria:

1) Test:

Is there a unique key index for all tables that includes a column or set of columns not including the primary key?

Procedure:

Look at the database metadata to ensure that each table has a unique key, and that the columns in the unique key are not also part of the primary key.

Example:

None

BP1258

Explicitly define the encoding style of all data transferred via XML.

Rationale:

By default, **XML** is encoded using **Unicode**. Consequently, data transferred via XML should explicitly specify the encoding style. Assuming the default can cause **interoperability** problems between implementations.

> ***Note:*** *Look for the following XML tag as the first line returned from queries that return XML from the database:*
>
> ```
> <?xml version="1.0" encoding="UTF-8"?>
> ```

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Syntax
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Syntax
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Syntax
NESI / Part 5: Developer Guidance / Data / XML / XML Syntax
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

BP1259

Use indexes.

Rationale:

An index in an **RDBMS** is a summary of information organized to minimize the search time. Indexes summarize the information in a table. So, an employee table might have an index of last names, or last name and first name.

Having additional indexes on tables involves a tradeoff between query performance and insert/update/delete performance, which requires underlying index maintenance.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

BP1260

Define a primary key for all tables.

Rationale:

By definition, a **primary key** uniquely defines each row within a table. To optimize the use of the table and to find records by the primary key, there should be an index that enforces the uniqueness of the key.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

Evaluation Criteria:

1) Test:

Is there a primary key defined for each table listed in the database?

Procedure:

Examine the database metadata to ensure there is a primary key for each table in the database.

Example:

None

BP1261

Monitor and tune indexes according to the response time during normal operations in the production environment.

Rationale:

Index efficiency depends on the data being indexed. Common variables follow:

- A sparsely populated table versus a densely populated table
- Data added in an presorted order versus a random order

Consequently, as the data changes, the efficiency of the index changes.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

BP1262

In the case of Oracle, define indexes against the foreign keys (FK) columns to avoid contention and locking issues.

Rationale:

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

BP1263

Gather storage requirements in the planning phase, and then allocate twice the estimated storage space.

Rationale:

Storage space on the disk always poses a problem for databases, so it is necessary to plan storage space carefully.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

BP1264

For high availability, use hardware solutions when geographic proximity permits.

Rationale:

There are many ways to achieve high availability. Some are based on hardware and others on software. As a general rule, hardware solutions use simple redundancy and are consequently less complex and fragile. If geographic proximity is not an issue, the hardware solution is preferable.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Relational Database Management Systems
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Relational Database Management Systems
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Relational Database Management Systems
NESI / Part 5: Developer Guidance / Data / Relational Database Management Systems

BP1265

Validate XML documents during document generation.

Rationale:

All **XML** passed between two systems or services must be valid. The XML document generator is responsible for ensuring that the document is valid and **well-formed**. If there are problems, the document generator is the only user that can effectively change the document.

Validity is checked via the use of a **W3C** Standard Validating parser. These parsers are built into most XML editors but are also available as stand alone products. Either the XML is valid or diagnostics are returned indicating where the XML is invalid.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / XML Validation
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / XML Validation
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / XML Validation
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / XML Validation

Evaluation Criteria:

1) Test:

Are all the documents exported from the system or service valid and **well-formed**?

Procedure:

Capture all the documents and validate them, using an XML editor or stand alone XML validation tool.

Example:

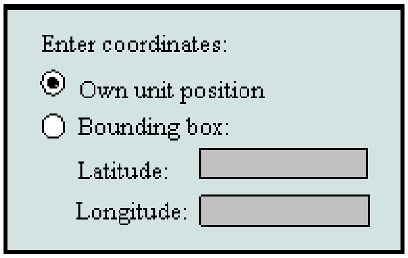
None.

BP1272

Disable dependent child controls when the parent control is inactive.

Rationale:

This practice makes it easier for the user to understand that the child controls depend on the selection of the parent, contributing to data integrity.



I1122

Referenced By:

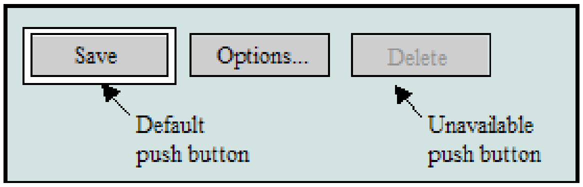
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

BP1273

Gray out the push button label if a button is unavailable.

Rationale:

This practice makes it easier for the user to understand that the button cannot be used until other action is taken.



I1126

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

BP1280

In tabular data displays, right justify integer data.

Rationale:

Whole numbers, displayed in a column, are easier to read if the digits of the same magnitude (1's, 10's, 100's, etc.) are vertically aligned.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Are all tabular whole number data right-justified?

Procedure:

Search all style sheets for the word "text-align." Examine the results for tabular whole number data and make sure the "text-align" attribute is set to "right"; visual Web page inspection may necessary to see if a defined align style is used within the tabular data.

Example:

Correct usage:

Cascading style sheet:

```
.td-items {
   text-align:right;
}
```

HTML:

Incorrect usage:

No alignment or incorrect alignment used.

BP1281

In tabular data displays, justify numeric data with decimals by using the decimal point.

Rationale:

It is common practice to align non-whole numbers by the decimal point for readability.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Are all tabular non-whole number data justified by decimal point?

Procedure:

Search all style sheets for the word "text-align." Examine the results for tabular non-whole number data and make sure the "text-align" attribute is set to "."; visual Web page inspection may be necessary to see if a defined align style is used within the tabular data.

Example:

Correct usage:

Cascading style sheet:

```
.td-subtotal {
   text-align:".";
}
```

HTML:

Incorrect usage:

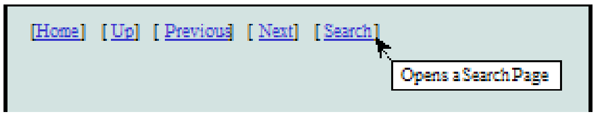
No alignment or incorrect alignment used.

BP1290

Use a tool tip to display help information about a control when the purpose of the control is not self-evident.

Rationale:

Using a tool tip increases user efficiency by preventing click errors. A mouse over event is the typical mapping for invoking a tool tip.



I1135

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

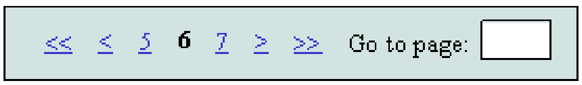
BP1291

Use obvious navigation controls for moving between pages in search results that span multiple pages.

Rationale:

Obvious navigation controls help a user to identify and use paging controls quickly. For example,

<	navigate back one page
>	navigate forward one page
<<	navigate back to the beginning page
>>	forward to the end page



I1136

Referenced By:

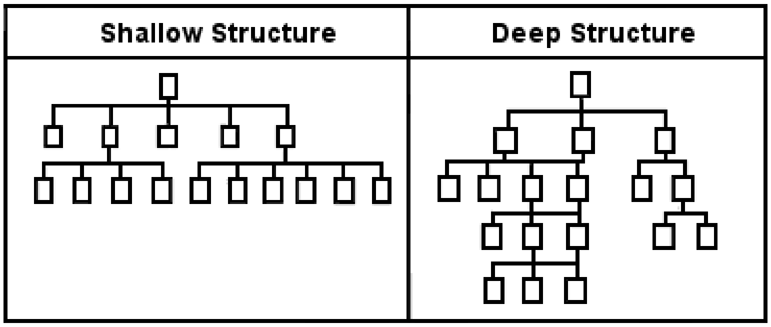
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

BP1297

Structure a Web site hierarchy so users can reach important information and/or frequently accessed functions in a maximum of three jumps.

Rationale:

Use a shallow structure rather than a deep structure. A user's success at finding a target drops off sharply after three clicks.



I1139

Referenced By:

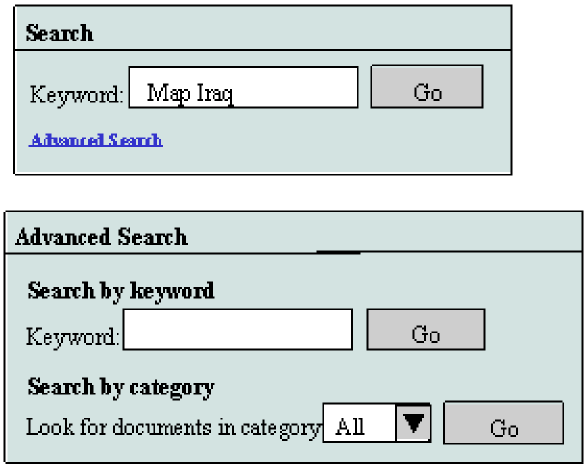
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces

BP1298

Provide basic search functionality as the default with a link or button that provides more advanced search features.

Rationale:

This practice makes the search feature cleaner and easier to use because the advanced features are hidden.



I1140

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

BP1299

Include a link back to the home page on all Web pages.

Rationale:

A link back to a Web site home page, for example in the form of a logo and a regular HTML link called **`Home`**, helps users navigate the Web site.

I1143

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces

BP1353

Use a data abstraction layer between the RDBMS and application for externally-visible applications to prevent the disclosure of sensitive data.

Rationale:

Large volume commercial online retailers often store customer data in an RDBMS, but they use a data abstraction layer with limited privileges to access that data from their Web services and other externally-visible applications. This more fully protects the data in the database from unauthorized access and modification.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security

Evaluation Criteria:

1) Test:

Does the application protect sensitive data by using a data abstraction layer between the application and RDBMS?

Procedure:

Check that sensitive data is not readable and modifiable externally by the application.

Example:

None

BP1355

Do not design the database around the requirements of an application.

Rationale:

Databases often outlive applications (i.e., legacy databases and evolution of applications). Database can also support multiple applications. If design of the database were around the application, it may present security holes that other applications could exploit. It is better to design the application around the rules set by the database.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / RDBMS Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / RDBMS Security

Evaluation Criteria:

1) Test:

Is application business logic or rules not found in the database?

Procedure:

Make sure data validation is done at database even if it is already being done at the application level.

Example:

None

BP1360

Use the XML Infoset standard to serialize messages.

Rationale:

XML signatures rely on a character-by-character comparison for proper operations. A one character difference is a different result. So using a standard for serialization is very important to successful communications.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security

Evaluation Criteria:

1) Test:

Does the Web service user serialize messages using the XML Infoset Standard?

Procedure:

Generate a test message and check it for compliance with the XML Infoset Standard.

Example:

None

2) Test:

Does the Web service provider serialize messages using the XML Infoset Standard?

Procedure:

Generate a test message and check it for compliance with the XML Infoset Standard.

Example:

None

BP1375

Use asymmetric encryption for sensitive SOAP-based Web services.

Rationale:

Most Web services exchange very few messages so the fact that asymmetric encryption is computationally intensive is a non-issue. Symmetric encryption is more efficient, but it is done by sharing a secret key outside the SOAP message communication which is less portable.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / SOAP Security
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Encryption and HAIPE
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Identity Management, Authentication, and Privileges
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Design Tenet: Identity Management, Authentication, and Privileges

BP1392

Register services in accordance with a documented service registration plan.

Rationale:

Program information services are provided via a shared space for use by consumers. In order to locate these services and access the corresponding information provided, the services should be registered in the **service registry** per direction of the shared information space manager.

Referenced By:

NESI / Part 2: Traceability / Naval Open Architecture / Reusability
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata
NESI / Part 5: Developer Guidance / Data / Metadata
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Accessible
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Design Tenet: Make Data Understandable
NESI / Part 2: Traceability / Naval Open Architecture / Interoperability
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Be Responsive to User Needs
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry

NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Has the Program generated default service definitions and registered them in the DoD service registry?

Procedure:

Review that there is a service definition (URLs, WSDL entries, etc.) for each of the program information services and that they have been registered accordingly.

Example:

None

BP1394

Identify, publish and validate data objects exposed to the enterprise early in the data engineering process and update in a spiral fashion as development proceeds.

Rationale:

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling

BP1396

Develop high-level conceptual data models for new systems prior to Milestone A based on the business process context in which the system will be used.

Rationale:

An early high-level understanding of the data objects/entities involved in a system can help to clarify the purpose and context of the system and identify potential downstream interoperability issues.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling

BP1397

Identify and develop use cases or reuse existing use cases as appropriate as early in the data engineering process as possible to support data model development.

Rationale:

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data-Centric Publish-Subscribe (DCPS) / Reading/Writing Objects within a DDS Domain
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling

BP1398

Develop Interaction models as appropriate.

Rationale:

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling

BP1400

Programs will use authoritative metadata established by the Joint Mission Threads (JMTs) when available.

Rationale:

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Joint Net-Centric Capabilities
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling

BP1408

Use a semantic description language such as Web Ontology Language (OWL) or Resource Definition Framework (RDF) to represent an Ontology.

Rationale:

Data producer recommendations are still maturing for how to handle data producers interaction with **Web Ontology Language** (**OWL**) or **Resource Definition Framework** (**RDF**).

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata
NESI / Part 5: Developer Guidance / Data / Metadata

BP1567

Use the <abbr> and <acronym> tags to specify the expansion of acronyms and abbreviations.

Rationale:

Provides the user with easy access to the meaning of abbreviations and acronyms.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients

BP1568

Use a markup language to represent mathematical equations within Web pages.

Rationale:

Use a markup language such as MathML to display equations rather than creating images to display equations. This provides a more semantic meaning to those who may want to parse and use the equation and also provides for a more maintainable display of the equation.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Browser-Based Clients
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Browser-Based Clients
NESI / Part 5: Developer Guidance / User Interfaces / Browser-Based Clients

BP1715

Design SCA log services according to the OMG Lightweight Log Service Specification.

Rationale:

One component of the SCA framework is a central logging facility, enabling the asynchronous collection of informational messages from any component connected to the framework; and the controlled read access to this information. The Lightweight Logging Service is a free-standing, self-contained service which is not connected to an event channel or similar infrastructure. Using a standard log service specification between SCA implementations can improve interoperability and portability.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: RF Acquisition
NESI / Part 2: Traceability / DISR Service Areas / Communications Applications / Software Communication Architecture
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Software Communication Architecture
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Communication Architecture
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Software Communication Architecture
NESI / Part 5: Developer Guidance / Middleware / Software Communication Architecture
NESI / Part 5: Developer Guidance / Logging

Evaluation Criteria:

1) Test:

Is the logging service designed according to the OMG Lightweight Log Service Specification? Is the logging service designed according to the OMG Lightweight Log Service Specification?

Procedure:

Check the log service provider's documentation for compliance with the OMG Lightweight Log Service Specification.

Example:

None.

BP1716

Develop applications for SCA-compliant systems using a higher order programming language.

Rationale:

Developing **Software Communications Architecture** (**SCA**) applications in higher order languages such as ***C*** enables independence from platform dependencies and helps ensure portability.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Communications Applications / Software Communication Architecture
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Software Communication Architecture
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Communication Architecture
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Software Communication Architecture
NESI / Part 5: Developer Guidance / Middleware / Software Communication Architecture

Evaluation Criteria:

1) Test:

Does the application use a higher order language such as C rather than a lower order language such as Assembly?

Procedure:

Check what programming language is used to develop the SCA application.

Example:

None.

BP1720

Do not use commonly predefined VHDL identifier names for other identifiers.

Rationale:

The use of predefined identifiers causes confusion and some compilers and simulators have difficulty dealing with such identifiers. This reduces code portability.

Note: *This practice has been adapted from Cohen [R1114], section 2.1.1.2.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / VHDL / VHDL Coding and Design

Evaluation Criteria:

1) Test:

Are any of the following predefined identifier names used, including the identifiers in the Std and IEEE design libraries: FF, Time, Min, Ns, Ms, ACK, Real, Std, On?

Procedure:

Check all other identifiers and make sure they are not the names of any predefined identifiers.

Example:

None

BP1721

Define a VHDL package for closely related VHDL items that support an application function.

Rationale:

A package represents a **module** that allows the specification of groups of logically related declarations. Frequently used pieces of VHDL code are usually written in the form of components, functions, or procedures. These pieces are then placed into a package and compiled into the destination library. This technique allows code partitioning, code sharing, and code reuse.

> ***Note:*** *This practice has been adapted from Cohen [R1114], section 8.1, and Pedroni [R1113], section 10.2.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / VHDL / VHDL Coding and Design

Evaluation Criteria:

1) Test:

Do the packages contain functionally related components, functions and procedures?

Procedure:

Check the code and make sure all packages contain functionally related components, functions and procedures.

Example:

None

BP1722

Employ VHDL components for commonly used VHDL described circuits.

Rationale:

A component is a special piece of conventional code that allows the construction of hierarchical designs. In other words, by declaring a piece of code as a component, that code can then be used within another circuit. This is just an additional way of partitioning a design and promoting code reuse and composeability.

> ***Note:*** *This practice been adapted from Pedroni [R1113], section 10.3.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / VHDL / VHDL Coding and Design

Evaluation Criteria:

1) Test:

Are commonly used circuit modules described as components?

Procedure:

Check the code and make sure commonly used circuit modules are described as components.

Example:

None

BP1723

Do not use guarded signals.

Rationale:

Guarded signals are not synthesizable and not commonly used. Guarded signals reduce the readability of code because the guards and drivers are not collected.

> ***Note:*** *This practice has been adapted from Cohen [R1114], section 6.2.7.1.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / VHDL / VHDL Synthesizable Design

Evaluation Criteria:

1) Test:

Does the signal kind (e.g. register, bus) appear in a signal declaration?

Procedure:

Check the signal declaration to see if the signal kind is stated. If so, the signal declared is a guarded signal of the kind indicated.

Example:

None

BP1732

Follow the Upper Camel Case (UCC) naming convention for XML Type names.

Rationale:

The predominate style used by most programs or projects is to use the **Upper Camel Case** (**UCC**) for type names. Type names should be easy to differentiate from namespace prefixes and from attributes. Since the namespace prefix and the type name are separated by a non-whites character (i.e., the colon, :), it is easier to identify the type name from the namespace name if the type name follows the UCC.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Defining XML Types
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas

Evaluation Criteria:

1) Test:

Do type names follow the **Upper Camel Case** (**UCC**) naming convention?

Procedure:

Examine the schema definition and verify that the type names follow the Upper Camel Case (UCC) name convention.

Example:

```
<xsd:complexType
   name="MyType"
   . . .
</ xsd:complexType>
```

BP1733

Follow the Upper Camel Case (UCC) naming convention for XML element names.

Rationale:

The predominate style used by most programs or projects is to use the **Upper Camel Case** (**UCC**) for **XML element** names. Element names should be easily differentiable from namespace prefixes and from attributes. Since the namespace prefix and the element name are separated by a non-whites character (i.e., the colon, :), it is easier to identify the element name from the namespace name if the element name follows the UCC.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas

Evaluation Criteria:

1) Test:

Do element names follow the **Upper Camel Case** (**UCC**) naming convention?

Procedure:

Examine the schema definition and verify that the element names follow the Upper Camel Case (UCC) name convention.

Example:

None.

BP1734

Follow the Lower Camel Case (LCC) naming convention for XML attributes.

Rationale:

The predominate style used by most programs or projects is to use the **Lower Camel Case** (**LCC**) for **XML attribute** names. Attributes are part of an attribute list which is a set of name="value" expressions separated by whitespace. Therefore, it is easy to find the beginning of the attribute name.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Defining XML Schemas

Evaluation Criteria:

1) Test:

Do type names follow the **Lower Camel Case** (**LCC**) naming convention?

Procedure:

Examine the schema definition and verify that the type names follow the Lower Camel Case (LCC) name convention.

Example:

None.

BP1739

Use the xsd qualifying prefix for XML Schema namespace.

Rationale:

Syntactically there is no reason why the XML Schema namespace can not be given any qualifier. However, for readability on the part of humans, using the xsd qualifier is clear, precise, concise and widely accepted.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces

Evaluation Criteria:

1) Test:

Does the XML schema use the xsd prefix for the XMLSchema namespace?

Procedure:

Look for the use of the XMLSchema namespace declaration and verify that the prefix is xsd.

Example:

The following is an example of using the xsd prefix for the XML Schema namespace:

```
<xsd:schema>
```

BP1741

Do not provide a schema location in import statements in schemas.

Rationale:

An import statement allows schema components from other schemas to be added to the current schema. The added schema components are associated with a namespace defined in the import statement. The import statement provides for the imported schema to also be optionally associated with a location where the schema can be found. Associating a schema location with a namespace during the import is referred to as early binding. This locks the definition to a specific implementation.

The following example highlights these points:

Weather Station Schema Definition

A weather station is defined as a collection of sensors with definitions that are to-be-determined.

> ***Note:*** *The import of the http://www.Sensor.org without specifying the optional schema location.*

> ***Note:*** *The use of the dangling type SensorType for the element Sensor. SensorType is bound later to a schema definition.*

```
<?xml version="1.0"?>
<xsd:schema
    xmlns: xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace=http://www.WeatherStation.org
    xmlns: s="http://www.Sensor.org"
    elementFormDefault="qualified">
 <xsd:import namespace="http://www.Sensor.org"/>
 <xsd:element name="WeatherStation">
   <xsd:complexType>
     <xsd:sequence>
      <xsd:element
           name="Sensor"
           type="s:SensorType"
           maxOccurs="unbounded"/>
    </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Sensor Supplier Schema Definition

A sensor supplier creates a sensor specific definition for a sensor.

```
<?xml version="1.0"?>
<xsd:schema xmlns: xsd="http://www.w3.org/2001/XMLSchema"
                      targetNamespace="http://www.Sensor.org"
                      xmlns ="http://www.Sensor.org"
                      elementFormDefault="qualified">
   <xsd:simpleType name="SensorType">
     <xsd:restriction base="xsd:string">
         <xsd:enumeration value="barometer"/>
         <xsd:enumeration value="thermometer"/>
        <xsd:enumeration value="anenometer"/>
     </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Weather Station Instance Document

A weather station instance document is created which binds the sensor suppliers definition of a sensor to the weather station. This allows the definition of the sensor to change or the location of the sensor definition (i.e., xsd) to change independently of the definition of the weather station.

```
<?xml version="1.0"?>
<ws:WeatherStation
   Xmlns: ws="http://www.WeatherStation.org"
   xmlns: xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation=
      "http://www.WeatherStation.org WeatherStation.xsd
       http://www.SensorSupplier.org SensorSupplier.xsd">
     <ws:sensor>thermometer</ws:sensor>
     <ws:sensor>barometer</ws:sensor>
     <ws:sensor>anenometer</ws:sensor>
</ws:WeatherStation>
```

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces

Evaluation Criteria:

1) Test:

Does the schema definition provide location for the imported schemas?

Procedure:

Examine the schema definition and make sure the schemaLocation attribute is not used in the import statement.

Example:

```
<xsd:import
  namespace="http://www.Sensor.org"
  schemaLocation=#Sensor.xsd#
```

```
/>
```

BP1742

Use the xsi qualifying prefix for XML Schema instance namespace uses.

Rationale:

Syntactically there is no reason why the XML Schema instance namespace can not be given any qualifier. However, for readability on the part of humans, using the xsi qualifier is clear, precise, concise and widely accepted.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Schema Documents / Using XML Namespaces
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Instance Documents

Evaluation Criteria:

1) Test:

Does the schema use the xsd prefix for the XMLSchema instance namespace?

Procedure:

Look for the use of the XMLSchema instance namespace declaration and verify that the prefix is xsi.

Example:

The following is an example of using the xsi prefix for the XML Schema instance namespace:

```
<xsd:schema xmlns: xsi="http://www.w3.org/2001/XMLSchema-instance">
```

BP1743

Use .xml as the file extension for files that contain XML Instance Documents.

Rationale:

By using the .xml extension for XML Instance Documents that are not associated with an application that requires another file extension (e.g., html, xslt):

- Readily identifies the file as containing XML to users
- Associates the XML file with various tools that work with XML Documents (i.e., browsers, parsers, validators, etc.)

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / XML Semantics / XML Instance Documents
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Semantics / XML Instance Documents
NESI / Part 5: Developer Guidance / Data / XML / XML Semantics / XML Instance Documents

Evaluation Criteria:

1) Test:

Are there XML files that do not have the XML file extension or that are associated with specific applications?

Procedure:

Scan the files looking for files that contain XML that are not associated with an application. Examples of files that are associated with applications or services are .wsdl, .html, .htm and .xsl.

Example:

None.

BP1747

Use the xsl qualifying prefix for XSLT namespace.

Rationale:

Syntactically there is no reason why the XSLT namespace can not be given any qualifier. However, for readability on the part of humans, using the xsl qualifier is clear, precise, concise and widely accepted.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / XSLT
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / XSLT

Evaluation Criteria:

1) Test:

Does the schema use the xsl prefix for the XSLT namespace?

Procedure:

Look for the use of the XSLT namespace declaration and verify that the prefix is xsl. Make sure there is only one namespace associated with the Transform XSD: http://www.w3.org/1999/XSL/Transform

Example:

The following is an example of using the xsl prefix for the XSL Transform namespace:

```
<xsl:stylesheet
xmlns: xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0"
    xmlns: xalan="http://xml.apache.org/xalan"
    xmlns: my-ext="ext1"
    extension-element-prefixes="my-ext">
```

BP1748

Separate static content from transformational logic in XSLTs.

Rationale:

Static XML content is content is copied verbatim from a static source, either internally or externally. Internal static content usually is found within the same input stream as the XSLT content. External static content is obtained from a different input stream and often comes from files or from data returned from a service.

Separating the static content from the transform logic facilitates maintenance by reducing the risk of unexpected side effects during the maintenance. In other words, maintenance to the transformational logic is isolated from the content. Content modifications have no affect on the transformation logic.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / XSLT
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / XSLT

Evaluation Criteria:

1) Test:

Is static content imported using the xsl:copy element that selects a document?

Procedure:

Look for the intermixing of static content with the XSLT transform code.

Example:

None.

BP1749

Use xsl:include for including XSL transforms.

Rationale:

Xsl:include includes other transforms and assigns the same precedence to the imported nodes as the importing document. This is the preferred method for including entire XSL transforms to allow for composition of multiple transforms into one that is much bigger.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / XSLT
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / XSLT

Evaluation Criteria:

1) Test:

Procedure:

Example:

```
<xsl:include href="Guidance.xsl"/>
```

BP1750

Use xsl:import for reusing XSL code.

Rationale:

Since xsl:import includes other XSL code with a lower precedence than the importing document, it is best to just include small snippets of reusable XSL code. Also, xsl:import is inefficient versus xsl:include when dealing with large documents.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / XSLT
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / XSLT
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / XSLT

Evaluation Criteria:

1) Test:

Procedure:

Example:

```
<xsl:import href="Guidance.xsl"/>
```

BP1752

Place dynamic XML element data within an XML CDATA section.

Rationale:

The content of dynamic data can not be predicted and could contain the XML special reserved characters < and & or the other characters that may cause parse errors; it is best to embed this data within an XML Character Data (CDATA) section that is ignored by parsers.

The following is an example of the use of a CDATA section that contains source code. Since the code could contain the < or & characters and be runtime dependent, a parse error could occur at runtime.
Please refer to the following example:

```
<![CDATA[
Public bool lessThan (a,b)
{ if (a!= null && b!=null a < b ) then
  { return true;
  } // End if
  else
  { return false;
  } // End else
} // End lessThan
]]>
```

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Syntax
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Syntax
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Syntax
NESI / Part 5: Developer Guidance / Data / XML / XML Syntax

Evaluation Criteria:

1) Test:

Do Element Data sections that are dynamically generated or are provided by external data surround the Element Data within a CDATA section?

Procedure:

Look for areas within XML instance documents or XML schemas that are candidates for dynamic content that can not be expected to be under the control of the XML instance document generator.

Example:

The following is an example of the use of a CDATA block that contains source code. Since the code could contain the < or & characters, a parse error could occur at runtime.
Please refer to the following example:

```
<![CDATA[
Public bool lessThan (a,b)
{ if (a < b ) then
  { return 1;
  } // End if
else
  { return 0;
  } // End else
} // End lessThan
]]>
```

BP1757

Do not ignore namespace prefixes in XPath expressions.

Rationale:

Ignoring namespaces can have undesired consequences. Some namespaces can contain nodes (elements) with the same name that contain different data structures. Consequently, if names bypass the use of the associated namespace, runtime errors can occur when attempts to process nodes of differing types occur.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / XPath
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / XPath
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / XPath
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / XPath

Evaluation Criteria:

1) Test:

Do any XPath statements ignore namespaces?

Procedure:

Check for the existence of XPaths similar to the following:

```
//*[local-name()='location']
```

location is a node name defined in two different namespaces. For example, a geographic namespace may define location as latitude and longitude. It may also be defined in the display namespace as a x and y pixel coordinate.

Example:

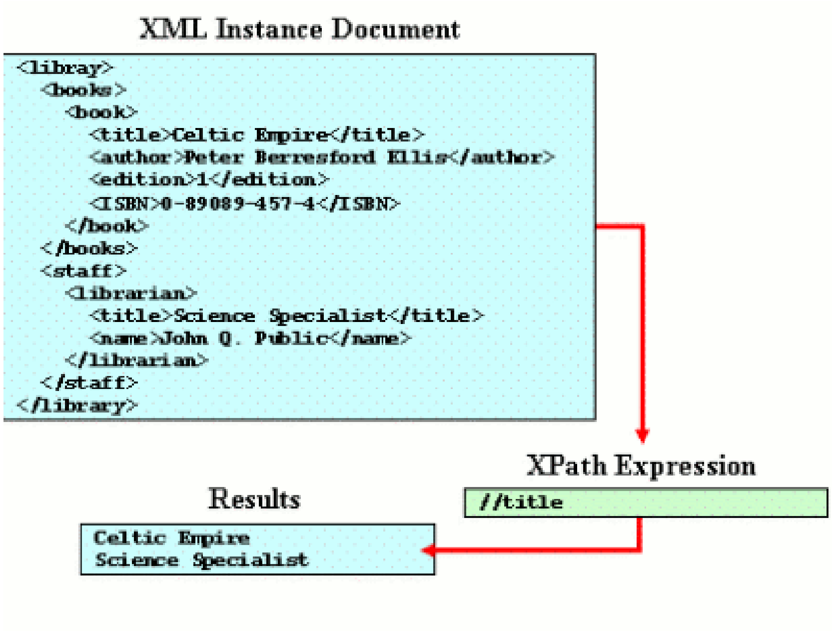
None.

BP1758

Make names in descendant expressions unique within an XML document.

Rationale:

The descendant operator, when misused, can have unintended consequences since nodes of the same name could possibly be included in multiple places in the XML Document. The XPath need to be written to eliminates unwanted nodes of the same name from other parts of the document.



I1172

In the above example, the <title> element can occur in multiple places within the document. Using the descendent operator '//' with the title element name returns all the titles.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / XPath
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / XPath
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / XPath
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / XPath

BP1764

Make all localizable user interface elements such as text and graphics externally configurable.

Rationale:

Externally configurable user interface elements allow for changing the supported language(s) at deploy-time or run-time without recompilation.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Designing User Interfaces for Internationalization
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Designing User Interfaces for Internationalization

Evaluation Criteria:

1) Test:

Are all localizable presentation elements such as user interface text and graphics externally configurable?

Procedure:

Check for external configuration files for localizable presentation user interface elements.

Example:

None.

BP1765

Declare the encoding type for all user interface content.

Rationale:

Declaring the encoding type allows for an application to determine the encoding type programmatically and make necessary display configuration settings at run-time. Also, for Unicode there are multiple ways to encode a character set.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Designing User Interfaces for Internationalization
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Designing User Interfaces for Internationalization

Evaluation Criteria:

1) Test:

Do the user interface components (such as HTML pages) declare the encoding type?

Procedure:

Check to see that user interface components declare the encoding type.

Example:

Send the charset parameter in the Content-Type of HTTP header:

```
Content-Type: text/html; charset=utf-8
```

For XML (including XHTML), use the encoding pseudo-attribute in the XML declaration at the start of a document:

For HTML or XHTML served as HTML, use the tag inside :

BP1766

Develop user interfaces to accommodate variable syntactic structure for messages.

Rationale:

Different languages form sentence structures in different ways. Composing messages in code from multiple substrings in order to display the messages to the user may cause problems when porting the code to a language that uses a different sentence structure.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Designing User Interfaces for Internationalization
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Designing User Interfaces for Internationalization
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Designing User Interfaces for Internationalization

Evaluation Criteria:

1) Test:

Are messages displayed on the user interface constructed in code using multiple substrings?

Procedure:

Check code for messages displayed to the user to see if the messages are composed from multiple substrings.

Example:

New.

BP1767

Follow a standards-based process for human systems integration engineering.

Rationale:

Using a standards-based process for human systems integration engineering, such as the that defined by the International Organization for Standardization in ISO 13407:1999 on human-centered design processes for interactive systems, increases the chance that required steps and procedures are completed during system development, leading to better usability.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction

Evaluation Criteria:

1) Test:

Was a process for human systems integration followed during system development?

Procedure:

Look for documentation stating the human systems integration process.

Example:

None.

BP1768

Use design patterns for application navigation.

Rationale:

Using common design patterns for application navigation builds on lessons learned, increases probability of user understand of the navigation pattern, and may result in better performance and a reduction in training.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / User Interface Services / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DISR Service Areas / User (Physical/Cognitive) / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Common End User Interfaces / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces
NESI / Part 5: Developer Guidance / User Interfaces / Human-Computer Interaction / Human Factor Considerations for Web-Based User Interfaces

Evaluation Criteria:

1) Test:

Does the application navigation follow design patterns?

Procedure:

Identify the design patterns used for application navigation.

Example:

- Use a hub navigation pattern for tasks that consist of multiple independent steps performed in any order
- Use wizard navigation pattern for tasks that consist of multiple interdependent steps that are defined in a predefined order.
- Use a pyramid navigation pattern when it is necessary to navigate to sibling, child, or parent pages while completing tasks.

BP1769

Provide wrapper or adapter classes to isolate XML parser implementations.

Rationale:

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / XML / XML Processing / Parsing XML
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / XML / XML Processing / Parsing XML
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / XML / XML Processing / Parsing XML
NESI / Part 5: Developer Guidance / Data / XML / XML Processing / Parsing XML

BP1780

Only overload arithmetic operators for objects that are arithmetic in nature.

Rationale:

In languages such as C++, it is possible to extend the intrinsic syntactical structure by defining overloaded operators. Operators that are naturally considered mathematical in nature (i.e., **`add`**, **`subtract`**, **`multiply`**, **`divide`**, etc.) should behave as expected. For example, if the addition operator **+** is defined, it should represent the mathematical addition operation.

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / C++ / C++ Operator Overloading

Evaluation Criteria:

1) Test:

Do overloaded mathematical operators perform any mathematical operations?

Procedure:

Review any mathematical operators that have been defined for any classes and ensure that they are mathematical in nature.

Example:

The following is an example of an addition operator:

```
class Imaginary
{ double value_;
  bool imaginary_;
  Imaginary
    ( double value,
      bool imaginary
    )
  { value_ = value;
    imaginary_ = imaginary;
  } // End Imaginary constructor
  Imaginary operator+
    ( Imaginary leftSideOfOperator )
  { ... // do what needs to be done
  } // End operator+
} // End Imaginary class
```

BP1781

Allocate and de-allocate all module objects within the module that contains the objects.

Rationale:

Sutter and Alexandrescu define a module as any cohesive unit of release maintained by a single person or team that is typically compiled with the same compiler, compiler version and compiler switches.

Because the memory allocation and de-allocation can change between these compiler instances, memory leaks and memory corruption can occur. Anytime memory allocation and de-allocation conflicts occur, there is a potential security issue.

> ***Note:*** *This practice has been adapted from Sutter and Alexandrescu [R1150], standard practice 60.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / C++ / C++ Namespaces and Modules

BP1782

Do not propagate exceptions across module boundaries.

Rationale:

Because the underlying definition of exceptions can vary between instances of a compiler, the resulting executable code could also vary resulting in not being able to properly communicate the exception.

> ***Note:*** *This practice has been adapted from Sutter and Alexandrescu [R1150], standard practice 62.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / C++ / C++ Namespaces and Modules

BP1783

Use portable types in a module's interface.

Rationale:

Because the types define the data that flows between modules and each compiler instance can vary these definitions, the types that define this data needs to be uniform in order to ensure proper data transfer.

> ***Note:*** *This practice has been adapted from Sutter and Alexandrescu [R1150], standard practice 63.*

Referenced By:

NESI / Part 5: Developer Guidance / Programming Languages / C++ / C++ Namespaces and Modules

BP1811

Isolate all use of vendor specific extensions to the Data Distribution Service (DDS).

Rationale:

Vendor specific extensions may be required to perform certain configuration actions, take advantage of features that are in the process of becoming standard (e.g., version 1.3, expected to be adopted by late 2007), or simply use additional capabilities provided by a vendor that would otherwise require significant application work.

Vendor-specific extensions should only be used if there is no standard API from the DDS specification that accomplishes the same function.
One method of isolating vendor-specific extensions is to enclose the code within conditional compile instructions (e.g., **`#ifdef #endif`** for C/C++) such that portability is not compromised.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe

Evaluation Criteria:

1) Test:

Does the implementation use wrappers or facade patterns to isolate vendor specific code?

Procedure:

Is vendor specific code contained within a limited number of classes or objects?

Example:

None

2) Test:

Does the implementation annotate vendor specific code?

Procedure:

Look for the use of compiler instructions that isolate vendor specific code.

Example:

```
#ifdef DDS_VENDOR_XXXX
…. <vendor specific code
```

```
#endif
```

BP1812

Use the RELIABILITY Quality of Service (QoS) kind BEST_EFFORT for Data Distribution Service (DDS) Topics that are written frequently where missing an update is not important because new updates occur soon thereafter.

Rationale:

The use of the **RELIABILITY** QoS kind **BEST_EFFORT** allows the middleware to use a lower-latency, lighter-weight protocol to send data that avoids the need for extraneous Acknowledgement and Heartbeat traffic. This protocol also exploit multicast more efficiently because there is never a need to send any acknowledgments back to the sender. Consequently, this protocol should be preferentially used whenever the nature of the Topic is such that occasionally missing a message has no adverse consequence to the system.

Data that is continually published and represents updates to data-objects or where only the most current value is of interest to the system are prime candidates for **BEST_EFFORT** communication.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Is the **RELIABILITY** QoS selection properly justified for each Topic? Is **BEST_EFFORT** kind used whenever the nature of the Topic allows it?

Procedure:

Review the system documentation for proper justification of the **RELIABILITY** QoS assigned to each Topic.

Example:

None

BP1813

Use the `RELIABILITY` Quality of Service (QoS) kind `RELIABLE` for Data Distribution Service (DDS) Topics written sporadically or where it is important that the current data in the Topic is received reliably.

Rationale:

The **`RELIABILITY`** QoS kind **`RELIABLE`** ensures the service will make all necessary attempts to deliver the information. The DDS protocol employs Heartbeats and Acknowledgments to accomplish this task.

Data that is rarely written or which the system requires never to be lost should be published with **`RELIABILITY`** QoS kind **`RELIABLE.`**

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Is the **`RELIABILITY`** QoS selection properly justified for each Topic? Is **`RELIABLE`** kind used whenever the nature of the Topic requires it?

Procedure:

Review the system documentation for proper justification of the **`RELIABILITY`** QoS assigned to each Topic.

Example:

None

BP1814

Use the `DEADLINE` Quality of Service (QoS) to for Data Distribution Service (DDS) DataWriters for which data is published at a constant rate.

Rationale:

The frequency with which a particular data-object is updated may affect the logic of the overall system. For example some radar processing algorithms may have been written under the assumption that each track is updated every five seconds after the radar completes a new sweep.

If the **`DataWriter`** specifies a **`DEADLINE`** QoS, DDS can monitor that each data-object is indeed written at least once per stated period. Furthermore, DDS can propagate the **`DataWriter`** deadline to the **DataReaders** such that they can realize whether their expectation matches what the **`DataWriter`** provides. If the expectation cannot be met the application is notified of an incompatible QoS.

By using this QoS the modules can remain de-coupled, yet provide the essential information required for the integrated system to operate as expected.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Is the **`DEADLINE`** QoS used in all the **`DataWriters`** where it could?

Procedure:

Review the system documentation for proper justification of the **`DEADLINE`** QoS assigned to each **`DataWriter`**.

Example:

None.

BP1815

Use the `DEADLINE` Quality of Service (QoS) for Data Distribution Service (DDS) DataReaders that expect data to be sent to them at a constant rate.

Rationale:

The frequency with which a particular data-object is updated may affect the logic of the overall system. For example some radar processing algorithms may have been written under the assumption that each track is updated every five seconds after the radar completes a new sweep.

If the **`DataReader`** specifies a **`DEADLINE`** QoS then DDS can monitor that an update to each data-object is indeed received at least once per stated period and if not notify the application. Furthermore, DDS can propagate the **`DataReader`** deadline to the DataWriters such that they can realize whether they can meet the expectation of the **`DataReader`**. If the expectation cannot be met the application is notified of an incompatible QoS.

By using this QoS the modules can remain decoupled, yet provide the essential information required for the integrated system to operate as expected.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Is the **`DEADLINE`** QoS used in all the **`DataReaders`** where it could?

Procedure:

Review the system documentation for proper justification of the **`DEADLINE`** QoS assigned to each **`DataReader`**.

Example:

None.

BP1816

Use the `LIVELINESS` Quality of Service (QoS) for Data Distribution Service (DDS) Topics where data is not sent sporadically; that is, it is sent with no fixed period.

Rationale:

Some data (e.g., alarms or commands) are sent without a fixed period. In these cases the fact that updates are not received could indicate that there is either no new data, or alternatively that there is a system malfunction and the writer is not able to send the data. The DDS **`LIVELINESS`** QoS allows the application to discern between these two situations.

Setting the **`LIVELINES`** QoS indicates to DDS that in the event that there is no data to send, periodic liveliness messages should be exchanged to notify the **DataReader** that the **DataWriter** is still active, capable of communication, and therefore that if it receives no data then it is in fact because there is none to send. The DDS monitors the **`LIVELINESS`** and informs the application when a **`DataWriter`** loses its ***liveliness*** via the proper status message dispatched to the Listener.

Proper settings of the **`LIVELINESS`** QoS is also required to receive proper **`InstanceState`** information with the received Samples as well as to manage **`OWNERSHIP`** in the presence of failures.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Are all **`DataWriters`** or **`DataReaders`** that do not set a **`DEADLINE`** setting a **`LIVELINESS`**?

Procedure:

Check the QoS used to create **`DataReaders`** and **`DataWriters`** and ensure that if the **`DEADLINE`** QoS is not set, then the **`LIVELINESS`** QoS is set to a non-infinite value

Example:

None.

BP1817

Use the `MANUAL_BY_TOPIC` setting of the `LIVELINESS` Quality of Service (QoS) for Data Distribution Service (DDS) Topics where the presence and health of the DataWriter is critical to the proper operation of the system.

Rationale:

Certain Topics are monitoring functions so critical to the health of the system that reliance on the health of the process that writes the Topic does not offer sufficient assurance that the application is performing the proper monitoring functions. In these situations the **`MANUAL_BY_TOPIC`** setting of the **`LIVELINESS`** QoS requires the **`DataWriter`** to either write the data at least once per liveliness period or invoke the **`DataWriterasset_liveliness()`** operation to indicate proper functioning.

The **`MANUAL_BY_TOPIC`** setting of the **`LIVELINESS`** QoS can be thought of as the distributed system equivalent to the mechanical dead man's switches used to monitor that the operator of a system (e.g., a train locomotive) is still present and able to function.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Are all critical **`DataWriters`** either setting a deadline or using a **`LIVELINESS`** set to **`MANUAL_BY_TOPIC`**?

Procedure:

Check the QoS used to create DataReaders and **`DataWriters`** and ensure that if the **`DEADLINE`** QoS is not set, then the **`LIVELINESS`** QoS is set to **`MANUAL_BY_TOPIC`** and has a non-infinite value.

Example:

None.

BP1818

Use the `HISTORY` Quality of Service (QoS) kind `KEEP_LAST` for Data Distribution Service (DDS) Topics that represent system state, in that new data-values replace the old values for each Keyed data-object.

Rationale:

Some Topics represent system state. The readers of the Topic need only know the most current value (or last set of N values) of each data-object published under the Topic. An example of this may be a Topic representing the reading of different temperature sensors. Applications only care to read the most recent value of each sensor. The same may be said of a Topic representing the expected arrival times of aircraft at a given airport.

The **`HISTORY`** QoS setting of **`KEEP_LAST`** indicates to the middleware that it should not attempt to store or propagate old values of data objects; instead, only the most recent value(s) are of interest. This allows DDS to conserve system resources (memory) as well as to save the bandwidth required to send information that is no longer relevant. Reader applications also benefit as they do not waste time reacting to data values that are no longer current.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Is the **`HISTORY`** QoS properly sent on all Topics?

Procedure:

Check the QoS used to create **DataReaders** and **DataWriters** and check how the **`HISTORY`** QoS is set. Ensure that a kind **`KEEP_LAST`** is used whenever the Topic represents system state.

Example:

None.

BP1819

Use the `HISTORY` Quality of Service (QoS) kind `KEEP_ALL` for Data Distribution Service (DDS) Topics that represent events or commands where all values written should be delivered to the readers (i.e., new values do not replace old values).

Rationale:

Some Topics represent events, commands, or messages in that new data written never replaces previously-written values, rather they should all be delivered to the **DataReader**.

The **`HISTORY`** QoS setting of **`KEEP_ALL`** indicates to the middleware that it should not replace old values with new values on the topic. Subject to other QoS (such as filters, ownership, lifespan) they should all be delivered to the **`DataReaders`**.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Is the **`HISTORY`** QoS properly sent on all Topics?

Procedure:

Check the QoS used to create **`DataReaders`** and **DataWriters** and check how the **`HISTORY`** QoS is set. Ensure that a kind **`KEEP_ALL`** is used whenever the Topic represents 'events', commands or messages.

Example:

None.

BP1820

Use `TIME_BASED_FILTER` Quality of Service (QoS) to protect DataReaders that cannot handle all the traffic that could be written by the writers on that Data Distribution Service (DDS)Topic and just need periodic updates on the most current data-values.

Rationale:

The **`TIME_BASED_FILTER`** QoS allows a **`DataReader`** to specify that it is interested only in (potentially) a subset of the values of the data. The filter states that the **`DataReader`** does not want to receive more than one value each minimum_separation, regardless of how fast the changes occur. The default setting is minimum_separation=0 indicating that the **`DataReader`** is potentially interested in all values.

In heterogeneous systems, it is common that some subsystems either cannot handle or do not choose to handle all the information available on a Topic. For example a high-level display at an airport control tower may not need to update the location of aircraft more often than each second as the human operators looking at the display would not be able to take advantage of faster refreshes. Nevertheless, the data is published at much higher rate to allow for algorithmic processing on other subsystems.

By setting the **`TIME_BASED_FILTER`** properly an application that has a well defined maximum refresh rate can protect itself from system reconfigurations which may result in a Topic being published faster than originally anticipated.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Is the **`TIME_BASED_FILTER`** QoS properly sent on all **`DataReaders`**?

Procedure:

Check the QoS used to create **`DataReaders`** and check whether the **`TIME_BASED_FILTER`** QoS is set. Ensure it is set to a proper non-zero minimum_separation whenever the application can be in a system where it is not expected to handle all the updates on the Topic.

Example:

None.

BP1821

Use the Data Distribution Service (DDS) `LIFESPAN` Quality of Service (QoS) to indicate that data is only valid for a finite time period and stale data is discarded after a certain expiration time elapses.

Rationale:

Some **Topics** represent data with a natural expiration. For example the location of an aircraft during flight becomes less relevant as the information ages and may not have any tactical value after a certain time elapses.

The setting of the **`LIFESPAN`** QoS indicates to DDS the maximum time duration during which the information is relevant. After this time elapses, DDS is no longer required to maintain the information or provide it to the **DataReaders**. Proper setting of this QoS can therefore save resources and bandwidth as well as save **`DataReaders`** from being notified of information that is no longer relevant.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Is the **`LIFESPAN`** QoS properly sent on all Topics?

Procedure:

Check the QoS used to create **DataWriters** and check whether the **`LIFESPAN`** QoS is set. Ensure it is set to a proper non-infinite duration whenever appropriate.

Example:

None.

BP1822

Use the `PARTITION` Quality of Service (QoS) to limit the scope of the data written/read on a Data Distribution Service (DDS) Topic to only the writer/readers that have a common partition.

Rationale:

The **`PARTITION`** QoS is used to introduce logical partitions within a Topic. A **DataWriter** only communicates with a **DataReader** if (in addition to matching the Topic and having compatible QoS) they share a common partition

The **`PARTITION`** QoS is set on the **Publisher** and **Subscriber** and affects all the **`DataWriters`** in the Publisher and **`DataReaders`** on the Subscriber.

The **`PARTITION`** QoS can be used to introduce a logical scope and the fact that it is adjustable at run-time makes it possible to perform system reconfigurations. For example, a **`DataReader`** could be temporarily isolated from the rest of the system by switching its Partition to something that nobody matches. Similarly a **`DataWriter`** and **`DataReader`** could be reconfigured to have an "isolated session" by switching to a partition that nobody else uses.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Is the **`PARTITION`** QoS used to simplify application logic where appropriates?

Procedure:

Check the QoS used to create Publisher and Subscriber and check whether the **`PARTITION`** QoS is used. Verify that the application does not use some other non-standard way to implement a use-case that could be supported using the **`PARTITION`** QoS.

Example:

None.

BP1823

Use the Data Distribution Service (DDS) `RESOURCES_LIMITS` Quality of Service (QoS) in platforms with limited memory or in real-time systems to properly configure the resources that will be utilized and avoid exhaustion of system resources at run-time.

Rationale:

The **`RESOURCE_LIMITS`** QoS on the **DataWriter** and **DataReader** specifies the resources that DDS can consume in order to meet the requested QoS.

While these limits can be left to their default "auto-grow" settings proper configuration of these limits is important in any system that has limited resources and is expected to operate reliably for long time spans. By setting the limits the developer can balance the resources consumed for each topic and protect the system against a mis-configuration when a **Topic** that produces too much data exhausts the resources needed to manage other Topics. This is especially important if other QoS do not limit the amount of data that the system would need to store (e.g. if **`HISTORY`** is set to **`KEEP_ALL`** and **`LIFESPAN`** is set to infinite).

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Is the **`RESOURCE_LIMITS`** QoS set on the **`DataWriter`** and **`DataReader`**?

Procedure:

Check the QoS used to create **`DataWriters`** and **`DataReaders`** and check whether the **`RESOURCE_LIMITS`** are set to some finite limits. Ensure that any **`DataWriters`** and **`DataReaders`** that have if **`HISTORY`** kind **`KEEP_ALL`** and **`LIFESPAN`** duration set to infinite use the **`RESOURCE_LIMITS`** to control the maximum resource utilization.

Example:

None.

BP1824

Use the `USER_DATA` Quality of Service (QoS) to communicate metadata on the DomainParticipant that may be used to authenticate the application trying to join the Data Distribution Service (DDS) Domain.

Rationale:

In many cases the application needs to send additional information that describes the **`DomainParticipant`** to other participants in the DDS Domain. This information can be used to authenticate the participant or to meet any other application-specific need.

The **`USER_DATA`** QoS on the **`DomainParticipant`** allows the application to store un-interpreted bytes that will be propagated via the DDS built-in discovery mechanism and will be accessible to the other **`DomainParticipants`** on the system.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Is the **`USER_DATA`** QoS set on the **`DomainParticipant`**?

Procedure:

Check the creation of the **`DomainParticipant`** and determine whether the **`USER_DATA`** QoS is used. Ensure that the application does not use another non-standard way to accomplish the same function.

Example:

None.

BP1825

Use the `ignore_participant` operation on the DomainParticipant to deny access to another DomainParticipant trying to join a Data Distribution Service (DDS) Domain.

Rationale:

The **`ignore_participant`** operation can be used by a **`DomainParticipant`** to prevent another **`DomainParticipant`** from communicating with the first participant. In combination with the **`USER_DATA`** QoS on the participant this mechanism can be used to authenticate **`DomainParticipants`**.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe

Evaluation Criteria:

1) Test:

Is the **`ignore_participant`** operation used whenever there is a requirement to prevent arbitrary participants from accessing the information the first participant publishes or subscribes?

Procedure:

Check the code for any occurrences of the **`ignore_participant`** operation.
Ensure that the application does not use another non-standard way to accomplish the same function.

Example:

None.

BP1826

Use the `USER_DATA` Quality of Service (QoS) on the DataWriters and DataReaders to communicate metadata that may provide application-specific information of the entity writing/reading data in a Data Distribution Service (DDS) Domain.

Rationale:

In many cases the application needs to send additional information that describes the **`DataWriter`** or the DataReader to other entities in the DDS Domain. This information can be used to authenticate the **`DataWriter`**/ Reader or to meet any other application-specific need.

The **`USER`**_DATA QoS on the **`DataWriter`** and the **`DataReader`** allows the application to store un-interpreted bytes that will be propagated via DDS's built-in discovery mechanism and will be accessible to the other **`DataWriters`** and **`DataReaders`** on the system.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

Evaluation Criteria:

1) Test:

Is the **`USER_DATA`** QoS set on the **`DataWriter`** and **`DataReader`**?

Procedure:

Check the creation of the **`DataWriter`** and **`DataReader`** and determine whether the **`USER_DATA`** QoS is used. Ensure that the application does not use another non-standard way to accomplish the same function.

Example:

None.

BP1827

Use the `ignore_publication` and `ignore_subscription` on the DomainParticipant to deny access to a Data Distribution Service (DDS) Topic by a specific DataWriter or DataReader.

Rationale:

The ignore_publication and ignore_subscription operation can be used by a **`DomainParticipant`** to prevent a **`DataWriter`** or **`DataReader`** from communicating with the entities in the participant. In combination with the **`USER_DATA`** QoS on the **`DataWriter`** and **`DataReader`** this mechanism can be used to check that the **`DataWriter`** and **`DataReader`** have the proper **access control** to the Topic.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe

Evaluation Criteria:

1) Test:

Are the **`ignore_publication`** and **`ignore_subscription`** operation used whenever there is a requirement to prevent arbitrary **`DataWriters`** or **`DataReaders`** from accessing the information on a Topic?

Procedure:

Check the code for any occurrences of the **`ignore_publication`** and **`ignore_subscription`** operation. Ensure that the application does not use another non-standard way to accomplish the same function.

Example:

None.

BP1828

Use the Data Distribution Service (DDS) `OWNERSHIP` Quality of Service (QoS) kind set to `SHARED` when each unique data-object within a DDS Topic to which multiple DataWriters can write.

Rationale:

A primary intent of DDS is to support a loosely coupled publish and subscribe paradigm where the publishing is isolated from subscribing through autonomous topics. As a result, an implementation that requires a single data publisher currently may evolve to require multiple data publishers in the future. By using a **`OWNERSHIP`** QoS kind set to **`SHARED`** and allowing the DDS infrastructure to connect the **publisher** and the **subscriber** together, the implementation may be extended to another DDS profile without having to modify the original source code.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

BP1829

Use the Data Distribution Service (DDS) `OWNERSHIP` Quality of Service (QoS) kind set to `EXCLUSIVE` when multiple DataWriters cannot write each unique data-object within a DDS Topic simultaneously.

Rationale:

DDS easily supports multiple **publishers** adding data to the same topic without impacting the **subscribers**. Using the DDS **`OWNERSHIP`** QoS kind set to **`EXCLUSIVE`** places the entire burden off supporting the multiple publishers on the DDS implementation rather than the publisher or subscriber code. This results in an increase of modularity, portability and the maintainability.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Layering and Modularity
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Quality of Service
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Quality of Service

BP1830

Use the Data Distribution Service (DDS) Content Profile to tailor subscription message data.

Rationale:

The DDS Content Profile allows for the **subscribers** to select and refine the data that is retrieved from a **Topic**. This tailoring code is part of the DDS infrastructure and is well tested and reliable. Not using the DDS Content Profile and using code within the subscriber increases the complexity of the subscriber and causes tight coupling between the subscriber code and the Topic.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Network Connectivity
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe

BP1831

Use the Data Distribution Service (DDS) Persistence Profile to ensure durable data delivery.

Rationale:

The DDS Persistence Profile allows for data persistence within a **Topic** independent of hardware platform and operating system (OS) and to retrieve the data using the standard **Structured Query Language** (SQL). As a result, the publisher, subscriber and the topic remain loosely coupled from each other as well as the hardware platform or the OS.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / Decoupling Using DDS and Publish-Subscribe

BP1832

Handle all Data Distribution Service (DDS) Data Local Reconstruction Layer (DLRL) Exceptions.

Rationale:

The DLRL API may raise Exceptions under certain conditions. The following is an extensive list of all possible Exceptions and the conditions in which they will be raised.

`DCPSError`	If an unexpected error occurred in the DCPS
`BadHomeDefinition`	If a registered **`ObjectHome`** has dependencies to other, unregistered **`ObjectHomes`**.
`NotFound`	If a reference is encountered to an object that has not (yet) been received by the **DCPS**.
`AlreadyExisting`	If a new object is created using an identify that is already in use by another object.
`AlreadyDeleted`	If an operation is invoked on an object that has already been deleted
`PreconditionNotMet`	If a precondition for this operation has not (yet) been met.
`NoSuchElement`	If an attempt is made to retrieve a non-existing element from a Collection.
`SQLError`	If an **SQL** expression has bad syntax, addresses non-existing fields or is not consistent with its parameters.

Note: *DLRL, a recent addition to the DDS specification is particularly rich; implementations using this upper level profile of the specification are still emerging.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)

BP1833

Use the Data Distribution Service (DDS) Object Model Profile for accessing message data as objects.

Rationale:

The DDS **Data Local Reconstruction Layer** (DLRL) is intended to provide an abstraction layer between the actual underlying data and the higher level object level concepts used in applications. The Object Model Profile defines how applications interact with the abstract object layer. Applications that are bound directly to the actual underlying data are tightly coupled to the layer and are subject to its evolutionary changes.

Note: *DLRL, a recent addition to the DDS specification is particularly rich; implementations using this upper level profile of the specification are still emerging.*

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Messaging / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Messaging / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)
NESI / Part 5: Developer Guidance / Middleware / Messaging / Data Distribution Service (DDS) / DDS Data Local Reconstruction Layer (DLRL)

BP1834

Develop software to operate in IPv4-only environments.

Rationale:

After migration to support **IPv6**, applications will still need to communicate with legacy **IPv4** applications and services. Maintaining compatibility ensures that the application remains portable even in environments that only support IPv4.

Referenced By:

NESI / Part 5: Developer Guidance / Source Code Migration to Support IPv4 and IPv6

Evaluation Criteria:

1) Test:

Does the application operate correctly in IPv4-only environments?

Procedure:

Test compatibility of the application in IPv4-only environments services.

Example:

None.

BP1855

Identify types of data items for potential sharing external to the program.

Rationale:

Identifying the types of data items that may be shared external to the program will drive the refinement of interoperability requirements and the design of interoperability mechanisms. Potential sources for this information include descriptions of existing data stores and existing or planned interfaces, architectural products, data models, document repositories, etc. Consider the logical entities represented by the data. Consider issues related to security classification, frequency of exchange, and file formats. Consider issues related to timeliness and data quality.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

BP1856

Identify specific data items for potential sharing external to the program.

Rationale:

Identifying the specific data items that may be shared external to the program will drive the refinement of interoperability requirements and the design of interoperability mechanisms. Potential sources for this information include descriptions of existing data stores and existing or planned interfaces, architectural products, data models, document repositories, etc. Identify the source, typical destinations, security classification, frequency of exchange, and typical size of the data. Avoid sharing data from other sources as a "pass through."

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

BP1857

Prioritize data items for potential sharing external to the program.

Rationale:

Prioritizing data items for potential sharing external to the program will support the planning of the migration to include the allocation of development resources. Analyze key operational processes to identify operationally important information exchanges. Consult with **Communities of Interest** (**COIs**) to determine the demand for specific data assets. Consider such factors as cost, time, and engineering difficulty.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

BP1858

Publish preliminary program data-related development plans.

Rationale:

While initially incomplete, preliminary program data-related development plans may prove useful to other programs as they plan their migrations due to the inherent interdependencies introduced by the Net-Centric Data Strategy. Create initial descriptions of data items that are forecast to be sharable using the **DoD Discovery Metadata Specification** (**DDMS**) and publish them in the **DoD Metadata Registry**.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

BP1859

Create external representations for sharable data items.

Rationale:

External representations will drive the implementation of both providers and consumers of the data items. Coordinate both internally within the program and externally with appropriate **COI**s. Explore de facto loose coupler and existing COI data formats. Create **XML schema** definitions for the data items and publish them in the **DoD Metadata Registry**.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

BP1860

Create metadata representations for sharable data items.

Rationale:

Metadata representations will drive the implementation of both providers and consumers of the data items. Identify what data items will be searchable taking into account cost and performance considerations. Tag individual data items as appropriate using automated metadata generation where possible. Represent discovery metadata using the **DoD Discovery Metadata Specification** (**DDMS**).

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

BP1861

Publish data access services that implement interfaces to shared data.

Rationale:

Services make data accessible using standardized mechanisms and enable the loose coupling of systems that process data. Select the appropriate underlying SOA-based technologies using NESI. Design service interfaces using the **XML schema** definition for the data exchange. Take into account security, performance, and versioning considerations. Use the **DoD Discovery Metadata Specification** (**DDMS**) and the **DoD Metadata Registry**. Test, deploy, and sustain data exchange mechanisms that support the NCDS in much the same fashion as any other mission-oriented software. The standard lifecycle methodologies used for other systems and software will apply.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Services
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Services
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Services
NESI / Part 4: Node Guidance / Services

BP1863

Make shareable data assets visible, even if they are not accessible.

Rationale:

Making data visible using a consistent, standardized metadata specification within a Net-Centric Environment (NCE) facilitates a federated cross-organizational discovery capability [R1172]. A common specification for the description of information allows for a comprehensive capability that can locate all information across the NCE regardless of format, type, location, or classification, dependent on user authorization. The **DoD Metadata Specification** (**DDMS**) was developed to support Enterprise-wide data discovery by providing a common set of descriptive metadata elements. Discovery metadata must conform to the DDMS in accordance with DoD Directive (DoDD) 8320.2 [R1217]. Information owners tag information with DDMS-compliant metadata to ensure discoverability of information in the NCE.

The extensible nature of the DDMS supports domain-specific or **COI** discovery metadata requirements and extends the element categories identified in the DDMS Core Layer used to describe information. Use of the DDMS does not preclude use of other metadata processes or standards. For example, record-level database tagging and in-line document tagging are common practices to support various department objectives. These tagging initiatives should be enhanced to include the DDMS for enterprise discovery.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: IPv6
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Service-Oriented Architecture (SOA)
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Open Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata Registry
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Visibility - Discoverable / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Policy / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Accessibility - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata Registry
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / Metadata Registry
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata Registry
NESI / Part 5: Developer Guidance / Data / Metadata Registry

Evaluation Criteria:

1) Test:

Does the system provide discovery metadata in accordance with the DoD Discovery Metadata Standard (DDMS) for all data posted to shared spaces?

Procedure:

Examine the DoD Metadata Registry for program/system.

Example:

Discoverable information has associated DDMS metadata that can be found in the DDMS).

BP1865

Provide sufficient program, project, or initiative metadata descriptions and automated support to enable mediation and translation of the data between interfaces.

Rationale:

Information exchanges should support known and unanticipated users. The program or project should initiate sufficient metadata descriptions and provide automated support to enable mediation and translation of data between interfaces. All of the data that can and should be shared externally beyond the programmatic bounds of your program should be defined well enough in metadata descriptions and translation of the data between interfaces should be automated.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Services / Core Enterprise Services (CES) / NCES Federated Search
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Services / Core Enterprise Services (CES) / NCES Federated Search
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Services / Core Enterprise Services (CES) / NCES Federated Search
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Accessibility - Operational / NCES Federated Search
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Discovery Services / NCES Federated Search
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Provide Core Enterprise Services / Core Enterprise Services (CES) / NCES Federated Search
NESI / Part 4: Node Guidance / Services / Core Enterprise Services (CES) / NCES Federated Search
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Evolve Computing Infrastructure / General Responsibilities / Coordination of Node and Enterprise Services
NESI / Part 4: Node Guidance / General Responsibilities / Coordination of Node and Enterprise Services
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Metadata
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Metadata
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Metadata
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - Registered / Metadata
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Metadata
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Metadata
NESI / Part 5: Developer Guidance / Data / Metadata
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Provide Data Management
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Interoperable
NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Net-Centric Information Engineering
NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Net-Centric Information Engineering
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Evolve Computing Infrastructure / General Responsibilities / Net-Centric Information Engineering
NESI / Part 4: Node Guidance / General Responsibilities / Net-Centric Information Engineering
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Data / Design Tenet: Make Data Visible
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Visibility / Design Tenet: Make Data Visible

Evaluation Criteria:

1) Test:

Evaluation of interfaces and applicable mediation/translations to access that the program, project, or initiative has sufficient metadata descriptions and automated support to enable mediation and translation of the data between interfaces. Data is XML wrapped for exchange and configured to support standard transactions with headers, trailers and bodies.

Procedure:

Evaluate the degree to which data is XML wrapped for exchange and configured to support standard transactions with headers, trailers and bodies.

Evaluation of the DoD Metadata Registry entries to assess sufficient metadata descriptions and automated support the enables mediation and translation of the data between interfaces.

Example:

XML wrapped data are intend for exchange, that is configured in terms of standard transactions with headers, trailers and bodies.

BP1868

Incorporate mechanisms to enhance Computing Infrastructure (CI) availability.

Rationale:

Computing Infrastructure (CI) must be survivable, resilient, redundant, and reliable in the presence of attacks, failures, accidents, and natural or man-made disasters. A robust CI must incorporate survivability, resiliency, redundancy, and reliability to ensure operational availability in support of information sharing in DoD, as well as externally with federal agencies, state and local governments, allies, and coalition partners. In the context of the CI, the measure of reliability is included as a critical element in ensuring high mean time between failures (MTBF).

Survivable: Survivability ensures that CI systems, subsystems, equipment, processes, procedures, or CI-related doctrine, organization, training, materiel, leadership, personnel, facilities (DOTMLPF) continue to fulfill critical mission requirements in the presence of attacks, failures, accidents, and natural or man-made disasters.

Resilient: Incorporation of resiliency into CI ensures the ability to automatically recover from, or adjust to, attacks, failures, or accidents. Fault tolerance is a key example of resilience that measures the ability to respond gracefully to an unexpected CI system, subsystem, process, or procedure failure.

Redundant: Incorporation of automatic redundancy into the CI ensures that alternative devices are available to perform the required system functionality if a primary device fails. Redundancy also ensures that system data remains accessible and corruption free when CI components fail.

Reliable: Reliable OS platforms, other software infrastructure, and hardware components are critical to ensuring that operators can depend on their ability to support system functions and applications. Bandwidth conservation mechanisms minimize latency and jitter, as well as the instability that comes from running processors and networks with high loads. Processing efficiency mechanisms, such as efficient software implementation techniques, allow applications to meet performance and latency requirements. Typically, reliability is measured in mean time between user failures (MTBUF). MTBF of CI components is one factor affecting the overall system MTBF.

A Continuity of Operations Plan (COOP) and disaster recovery planning are also key to ensuring a robust CI. The DoD Dictionary of Military Terms defines COOP as "the degree or state of being continuous in the conduct of functions, tasks, or duties necessary to accomplish a military action or mission in carrying out the national military strategy." It includes the functions and duties of the commander, as well as the supporting functions and duties performed by the staff and others acting under the authority and direction of the commander.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Availability
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Develop and Implement Computing Infrastructure / Node Computing Infrastructure / Host Information Assurance
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Node Computing Infrastructure / Host Information Assurance
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Allocate Computing Infrastructure Resources / Node Computing Infrastructure / Host Information Assurance
NESI / Part 4: Node Guidance / Node Computing Infrastructure / Host Information Assurance
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Integrity / Computing Infrastructure Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Integrity / Computing Infrastructure Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Integrity / Computing Infrastructure Integrity

NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Integrity / Computing Infrastructure Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Integrity / Computing Infrastructure Integrity
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Integrity / Computing Infrastructure Integrity
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Enterprise Security / Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Network Resource Management Mechanism Protection / Security and Management / Enterprise Security / Integrity
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Enterprise Security / Integrity
NESI / Part 4: Node Guidance / Security and Management / Enterprise Security / Integrity
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Services / Design Tenet: Enterprise Service Management

Evaluation Criteria:

1) Test:

Does the program or initiative have a Continuity of Operations Plan (COOP)?

Procedure:

Verify existence of COOP.

Example:

Continuity of Operations Plans and Disaster Recovery Plans that include preparatory measures, response actions, and restoration activities planned or taken to ensure continuation of critical functions to maintain effectiveness, readiness, and survivability.

Technologies that allow, self-correcting mechanisms to be implemented (e.g., automatic recovery without manual intervention).

Clustering of servers, incorporation of relative addressing schemata (e.g., **DNS**), site mirroring, and provisioning of geographically distributed CI functionality are examples of fail-over implementations.

BP1880

Justify, document, and obtain a waiver for all radio terminal acquisitions that are not JTRS/SCA compliant.

Rationale:

Tactical communications programs should focus on attaining the end objective of providing a family of software-programmable radios that will greatly enhance warfighters' wireless communication capabilities, while decreasing cost of ownership for infrastructure. The Joint Tactical Radio System (JTRS) will provide critical communications capabilities for the tactical wireless tails of the GIG. JTRS and its software communications architecture (SCA) continue to evolve and have become a cornerstone of the provision of future net-centric capabilities.

Referenced By:

NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Information Assurance/Security / Design Tenet: Employment of Wireless Technologies
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Concurrent Transport of Information Flows
NESI / Part 2: Traceability / DISR Service Areas / Communications Applications / Software Communication Architecture
NESI / Part 2: Traceability / DISR Service Areas / Environment Management / Software Communication Architecture
NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Communication Architecture
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Software Communication Architecture
NESI / Part 5: Developer Guidance / Middleware / Software Communication Architecture
NESI / Part 2: Traceability / ASD(NII): Net-Centric Guidance / Transport / Design Tenet: Joint Net-Centric Capabilities

Evaluation Criteria:

1) Test:

Are all of the program's, project's, or initiative's radio acquisitions JTRS/SCA compliant?

Procedure:

Describe all radio acquisitions that are not JTRS/SCA compliant.

Example:

None.

BP1881

Separate code based on required privilege.

Rationale:

Separating code based on privilege allows for each function, process, or executable to run with a minimal set of privileges.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Principle of Least Privilege
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Principle of Least Privilege
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Principle of Least Privilege
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Principle of Least Privilege
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Principle of Least Privilege
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Principle of Least Privilege

BP1888

Only enable plaintext viewing in email clients on DoD-owned and DoD-operated information systems.

Rationale:

Due to the significant risk of malicious mobile code downloaded into user workstations via email, DoD Mobile Code Policy restricts all mobile code in email independent of risk category. Disabling the automatic execution of mobile code in email is for both mobile code contained in the body of an email message and attachments. This will prevent immediate automatic execution of HTML that may download and execute mobile code from remote sites when the user clicks on a message to preview it. The user will be able to preview the message, optionally view the page source of suspicious-looking messages, and subsequently decide whether to open the attachment (the user will still be able to intentionally select the email attachment to execute HTML in that attachment.)

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Mobile Code
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Mobile Code

Evaluation Criteria:

1) Test:

Is automatic execution of all categories of mobile code in email disabled?

Procedure:

Verify that only plaintext email viewing is enabled.

Example:

None.

BP1889

Minimize execution at elevated privilege levels to the shortest time required.

Rationale:

Holding elevated permission for a minimum time reduces the chance that a security exploit can execute arbitrary code and minimizes the impact when an exploit occurs.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Principle of Least Privilege
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Principle of Least Privilege
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Principle of Least Privilege
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Principle of Least Privilege
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Principle of Least Privilege
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Principle of Least Privilege

BP1890

Compile code using the highest compiler warning level available.

Rationale:

Compiler warnings are often useful in detecting possible violations of syntax rules and mistakes introduced by developers which may lead to run time errors.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings

Evaluation Criteria:

1) Test:

Is code compiled using the highest compiler warning level available for the compiler?

Procedure:

Verify that the build script includes an applicable flag to enable the highest warning level for the compiler.

Example:

Java compilers version 5 and higher support a **`-Xlint`** compile option.

BP1891

Develop code such that it compiles without compiler warnings.

Rationale:

Compiler warnings are often useful in detecting possible violations of syntax rules and mistakes introduced by developers which may lead to run time errors.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings

BP1892

Explicitly document exceptions for valid code that produces compiler warnings.

Rationale:

It is important to document exceptions when valid code produces a compiler warning as it aids maintenance and documents the reason for the warning which is useful for future development of the code and peer reviews. Often the documentation method for a programming language will also allow for suppressing the compiler warning which prevents false positive warning in the compiler output.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Heed Compiler Warnings

BP1893

Return meaningful, but non-sensitive, information from exception handlers.

Rationale:

Purging or sanitizing exception shown to users reduces the risk of exposing information to a user that may be used to form an exploit.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Handle Exceptions

BP1898

Purchase computers which contain a Trusted Platform Module (TPM).

Rationale:

Supporting TPM is a desirable requirement at this time, since many **DoD** components want to leverage its capabilities in the future for the protection of **data at rest** (**DAR**) on mobile computing devices. TPM is readily available in the commercial marker, and in most cases is standard on new computers.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Data at Rest
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Data at Rest

BP1901

Use Universal Core (UCore) as the basis for information exchange models for systems that exchange internal data with external systems.

Rationale:

UCore defines a specification containing agreed-upon representations for the most commonly shared and universally understood concepts of "who," "what," "when" and "where." Chairman of the Joint Chiefs of Staff Instruction (CJCSI) 6212.01E, *Interoperability and Supportability of Information Technology and National Security Systems*[R1175] recommends using UCore; this use is consistent with the *DoD Net-Centric Data Strategy*.[R1172]

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Data Interchange Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Data Management Services / Data / Data Modeling
NESI / Part 2: Traceability / DISR Service Areas / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Data Exposure Verification Tracking Sheet / Data Understandability / Data Modeling
NESI / Part 2: Traceability / Exposure Verification Tracking Sheets / Service Exposure Verification Tracking Sheet / Service Understandability - COI Data Models / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Foster Development for Standard Semantics / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide for Globalization / Internationalization Services / Data Modeling
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Facilitate Computing Infrastructure Knowledge Management / Data / Data Modeling
NESI / Part 5: Developer Guidance / Data / Data Modeling

BP1903

Include an xsd:dateTime field within long-lived XML digital signatures.

Rationale:

Just as in hand-written signatures, the time of signing in important consideration in long-lived digital signatures.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Technologies and Standards for Implementing Software Security / XML Digital Signatures
NESI / Part 5: Developer Guidance / Software Security / Technologies and Standards for Implementing Software Security / XML Digital Signatures

Evaluation Criteria:

1) Test:

Does the XML digital signature contain a field of type **`xsd:dateTime`**?

Procedure:

Verify the XML digital signature contains a field of type **`xsd:dateTime`**.

Example:

None.

BP1908

Provide bidirectional mediation between transport protocols mandated in the Defense IT Standards Registry (DISR) when implementing an Enterprise Service Bus (ESB).

Rationale:

ESBs provide transport protocol agnostic messaging between service producers and consumers. ESBs use transport protocol mediation to achieve this goal. Service interactions are not simple, one-way activities, but require an interactive dialog between the service producer and the consumer. To achieve this dialog, all protocol mediation needs to be bi-directional. Suporting mediation for transport protocols specified by the DISR allows message proucers and consumers flexibility in choice of transport protocol.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Enterprise Service Bus (ESB)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Enterprise Service Bus (ESB)
NESI / Part 5: Developer Guidance / Middleware / Enterprise Service Bus (ESB)

BP1909

Provide for filtering of XML messages using XML Path Language (XPath) when implementing an Enterprise Service Bus (ESB).

Rationale:

ESBs provide filtering and restricting of messages in order to match messge producers and consumers. XPath is a language specifically intended for effectively and efficiently finding information within an XML document. Therefore, syntax and tools that are based on XPath are preferred filter methods for messages formulated as XML documents sent over an ESB.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Enterprise Service Bus (ESB)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Enterprise Service Bus (ESB)
NESI / Part 5: Developer Guidance / Middleware / Enterprise Service Bus (ESB)

BP1911

Provide for routing of messages based on message content when implementing an Enterprise Service Bus (ESB).

Rationale:

The ability to route messages based on message content allows for flexible dynamic matching of content producers and consumers.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Enterprise Service Bus (ESB)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Enterprise Service Bus (ESB)
NESI / Part 5: Developer Guidance / Middleware / Enterprise Service Bus (ESB)

BP1913

Provide for mediation between synchronous and asynchronous messages when implementing an Enterprise Service Bus (ESB).

Rationale:

ESBs support synchronous and asynchronous communication models. Allowing for mediation between synchronous consumers and asynchronous producers, and vice versa, allows for more flexible matching of message producers and consumers.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Distributed Computing Services / Enterprise Service Bus (ESB)
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Middleware / Enterprise Service Bus (ESB)
NESI / Part 5: Developer Guidance / Middleware / Enterprise Service Bus (ESB)

BP1914

Develop software to operate in dual stack environments.

Rationale:

It is not likely that software systems will be migrated to **IPv6** simultaneously. In order to interoperate with **IPv4** applications and services the software should operate in a dual stack mode during transition supporting both IPv4 and IPv6.

Referenced By:

NESI / Part 5: Developer Guidance / Source Code Migration to Support IPv4 and IPv6

Evaluation Criteria:

1) Test:

Is the application capable of supporting both IPv4 and IPv6 in dual stack environments?

Procedure:

Test for correct operation of the software in dual stack environments.

Example:

None.

BP1922

Design systems to have security as a core capability.

Rationale:

Adding non-functional capabilities, such as timeliness, fault management, and security, to a designed or implemented system usually is not cost-effective, if possible to do at all. Those capabilities are integral to the operation and thus significantly affect the design and implementation from the beginning of the initial modeling.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Practice Defense in Depth
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Practice Defense in Depth
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Practice Defense in Depth
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Practice Defense in Depth
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Practice Defense in Depth
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Practice Defense in Depth
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Develop and Implement Computing Infrastructure / Node Computing Infrastructure / Time-Critical Operations
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Net-Centric Environments / Time-Critical Operations
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Provide Computing Infrastructure Controls / Node Computing Infrastructure / Time-Critical Operations
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Computing Infrastructure Readiness / Allocate Computing Infrastructure Resources / Node Computing Infrastructure / Time-Critical Operations
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Communications Readiness / Support Quality of Service (QoS) Standards / Time-Critical Operations
NESI / Part 4: Node Guidance / Node Computing Infrastructure / Time-Critical Operations

BP1924

Develop software to be IP version agnostic.

Rationale:

To ensure maximum interoperability and portability, software should be built such that they are agnostic to the underlying IP version. This will maximize software compatibility while other network addressable resources transition to support **IPv4** and **IPv6**.

Referenced By:

NESI / Part 5: Developer Guidance / Source Code Migration to Support IPv4 and IPv6

Evaluation Criteria:

1) Test:

Does all software support both IPv4 and IPv6?

Procedure:

Test for correct operation of the software in dual stack environments.

Example:

None.

BP1925

Identify all IPv4 dependent code in source code.

Rationale:

It is necessary to identify parts of the source code that are **IPv4**-specific when porting applications to support **IPv6**.

Referenced By:

NESI / Part 5: Developer Guidance / Source Code Migration to Support IPv4 and IPv6

BP1926

Eliminate dependencies on a fixed IP address.

Rationale:

Dependencies on fixed IP addresses can create issues with **IPv6**. IPv6 addresses are much longer and more complex than **IPv4** addresses and there is no guarantee that the address will remain the same throughout the runtime of the software.

Referenced By:

NESI / Part 5: Developer Guidance / Source Code Migration to Support IPv4 and IPv6

Evaluation Criteria:

1) Test:

Does the application have any dependencies on an IP address being fixed?

Procedure:

Check for the usage of IP addresses and determine if the references are relying on the address remaining the same.

Example:

None.

BP1927

Use the `getaddrinfo()` function when resolving the address of an IP host.

Rationale:

The **`getaddrinfo()`** function is supported for both **IPv4** and **IPv6** environments.

Referenced By:

NESI / Part 5: Developer Guidance / Source Code Migration to Support IPv4 and IPv6

Evaluation Criteria:

1) Test:

Does the application use **`getaddrinfo()`** system call?

Procedure:

Look for the existence of **`getaddrinfo()`**.

Example:

None.

BP1928

Use the `getnameinfo()` function when getting the hostname of an IP address.

Rationale:

The **`getnameinfo()`** function is supported for both **IPv4** and **IPv6** environments.

Referenced By:

NESI / Part 5: Developer Guidance / Source Code Migration to Support IPv4 and IPv6

Evaluation Criteria:

1) Test:

Does the application use **`getnameinfo()`** function to resolve the hostname of an IP address?

Procedure:

Look for the existence of **`getnameinfo()`**.

Example:

None.

BP1929

Support the colon (`:`) in both IPv4 and IPv6 IP addresses.

Rationale:

IPv4 and **IPv6** use the colon (`:`) for different purposes. It is essential to correctly parse the colon within **IP** addresses for correct operation.

Referenced By:

NESI / Part 5: Developer Guidance / Source Code Migration to Support IPv4 and IPv6

Evaluation Criteria:

1) Test:

Does the application handle the colon (**`:`**) in IP addresses properly?

Procedure:

Look for anywhere IP addresses are parsed or generated by the application and ensure that it handles the colon (`:`) properly.

Example:

None.

BP1930

Identify network addressable resources using host names.

Rationale:

IP addresses can change during runtime, especially in **IPv6** environments. **IPv4** and IPv6 have different formats for IP addresses resulting in code complexity when using IP addresses compared to host names. A network addressable resource may have many IP addresses assigned to a given host name.

Referenced By:

NESI / Part 5: Developer Guidance / Source Code Migration to Support IPv4 and IPv6

Evaluation Criteria:

1) Test:

Does the application identify network addressable resources using host names?

Procedure:

Ensure that all network node addresses are identified as host names rather than IP addresses.

Example:

None.

BP1931

Use generic address structures for both IPv4 and IPv6 addresses.

Rationale:

Using generic addresses structures to store **IP** addresses enhances code portability and contributes to the source code being IP version agnostic.

Referenced By:

NESI / Part 5: Developer Guidance / Source Code Migration to Support IPv4 and IPv6

Evaluation Criteria:

1) Test:

Does the application use generic IP address structures that are compatible with IPv4 and IPv6 addresses?

Procedure:

Identify all IP address structures and ensure that each is IPv6 and IPv4 compatible.

Example:

None.

BP1932

Use an alternative to broadcast addresses in software systems using IPv6.

Rationale:

IPv6 does not have the broadcast addresses, so IPV6 software system redesign is necessary to use anycast and **multicast**. This can require a significant amount of work depending on an application's reliance on broadcast addresses.

Referenced By:

NESI / Part 5: Developer Guidance / Source Code Migration to Support IPv4 and IPv6

Evaluation Criteria:

1) Test:

Do software systems using IPv6 use anycast or multicast instead of broadcast addresses?

Procedure:

Verify that broadcast addresses are not used.

Example:

None.

BP1944

Peer review source code.

Rationale:

Code peer reviews are a useful tool to improve software quality and reduce defects (to include security related defects). Early identification of defects often reduces the total lifecycle cost by reducing the costs to repair the defects. Therefore, source code audits and peer reviews should be a regular activity during software development and maintenance along with normal testing activities. To the extent possible, utilize automated tools to assist in verifying that code meets standards as defined in the applicable coding standard document. This will result a more repeatable process and shorten the time required for a peer review.

Referenced By:

NESI / Part 2: Traceability / DISR Service Areas / Operating System Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Quality Assurance to Software Development
NESI / Part 2: Traceability / DISR Service Areas / Security Services / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Quality Assurance to Software Development
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Data and Services Deployment / Enable Trust / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Quality Assurance to Software Development
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Enclave, Network and Boundary Protection / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Quality Assurance to Software Development
NESI / Part 2: Traceability / DoD Information Enterprise Architecture / DoD Information Enterprise Architecture Activities / Provide Secured Availability / Provide Data in Transit and Data at Rest Protection / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Quality Assurance to Software Development
NESI / Part 5: Developer Guidance / Software Security / Policies and Processes for Implementing Software Security / Secure Coding and Implementation Practices / Apply Quality Assurance to Software Development

BP1948

Use a logging facade that supports timestamps.

Rationale:

Correct timestamps are necessary for understanding the sequence of events and for correlating events across multiple logs. The required degree of timestamp resolution may vary by system, but timestamp resolution must be of sufficient accuracy to provide value the consumer of the logs and to allow for correlation across logs. In most cases, log entries should contain timestamps with resolution at least accurate to the second.

Referenced By:

NESI / Part 5: Developer Guidance / Logging

BP1949

Write logging entries such that they are a single line in length.

Rationale:

In order to make logs easier to consume by people as well as computers, most logging implementations make use of plain text log entries. Modern systems are often comprised of multiple, often distributed, components where logging data is often spread across multiple locations. Log entries that are single line in length are easier to collate when combing logging data from multiple locations.

Single line log entries are also easier to search and filter by keword as the entire log entry is returned upon a match made on a particular log line. Sometimes it is not practical to log event data is a single line, such as in the case of logging stack traces.

Referenced By:

NESI / Part 5: Developer Guidance / Logging

BP1950

Use a logging facade that supports multiple logging levels.

Rationale:

Multiple logging levels are required by log consumers as log levels indicate the importance of log event information. Logging data to the appropriate logging levels allows for the filtering and searching of logging data by logging level.

Using multiple logging levels allows for the tuning of what information is logged; this helps balance the need for information vs. information overload, performance penalties, and storage issues associated with large logging volumes.

Referenced By:

NESI / Part 5: Developer Guidance / Logging

BP1951

Use a logging framework that supports log rotation.

Rationale:

Log rotation and truncation policies are best specified by the end user of a system through configuration. Modern logging frameworks provide for flexible configuration of log rotation and truncation.

Software developers should rely upon the underlying logging framwork to provide log rotation rather that write custom code to rotate logs which is often difficult and may not meet end users requirements.

Referenced By:

NESI / Part 5: Developer Guidance / Logging

BP1952

Use a logging facade that supports configurable output handlers.

Rationale:

Modern software systems are often composed of modular, resusable, software components. Allowing for the congfiguration of logging output handlers leads to increased source code interoperability and resusability while providing flexibility in configuration of the logging data final destination.

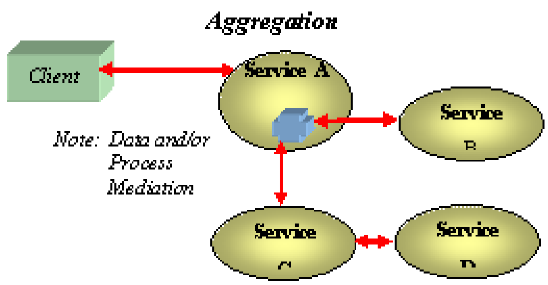
Referenced By:

NESI / Part 5: Developer Guidance / Logging

Glossary

.NET Framework		The .NET Framework is an integral Windows component that supports building and running the next generation of applications and XML Web services. The .NET Framework has two main components: the common language runtime and the .NET Framework class library. (Source: MSDN ***.NET Framework Conceptual Overview***, http://msdn.microsoft.com/en-us/library/zw4w595w.aspx)
Access Control		Limiting access to information system resources only to authorized users, programs, processes, or other systems. (Source: *National Information Assurance (IA) Glossary*, CNSSI 4009, revised June 2006) ***Note:*** *See also the following:* • ***Access Control List*** *(**ACL**) [GL1889]* • ***Discretionary Access Control*** *(**DAC**) [GL1197]* • ***Role-Based Access Control*** *(**RBAC**) [GL1643]*
Access Control List	ACL	In computer security, ACL is a concept used to enforce privilege separation. It is a means of determining the appropriate access rights to a given object depending on certain aspects of the process that is making the request, principally the process's user identity. In networking, ACL refers to a list of ports and services that are available on a host, each with a list of hosts and/or networks permitted to use the service. Both individual servers as well as routers can have access lists. Access lists are used to control both inbound and outbound traffic, and in this context they are similar to firewalls. (Source: http://en.wikipedia.org/wiki/Access_control_list)
Accredited Standards Committee X12	ASC X12	In 1979, the **American National Standards Institute** (**ANSI**) chartered the Accredited Standards Committee (ASC) X12 to develop uniform standards for interindustry electronic exchange of business transactions-electronic data interchange (**EDI**). (Source: http://www.x12.org/x12org/about/faqs.cfm#b1)
Active Directory	AD	An implementation of Lightweight Directory Access Protocol (LDAP) directory services by Microsoft for use in Windows environments; allows administrators to assign enterprise-wide policies, deploy programs to many computers, and apply critical updates to an entire organization. An Active Directory stores information and settings relating to an organization in a central, organized, accessible database. Active Directory networks can vary from a small installation with a few hundred objects, to a large installation with millions of objects. (Source: http://en.wikipedia.org/wiki/Active_Directory)

<table>
<tr><td>Adapter</td><td></td><td>An intermediary that translates between incompatible components interfaces, allowing them to communicate.</td></tr>
<tr><td>Aggregation</td><td></td><td>When information is derived from multiple sources a mediator service may aggregate the data and thus make many services appear to be one.



I1148

Note: See Mediation.</td></tr>
<tr><td>American National Standards Institute</td><td>ANSI</td><td>Administrator and coordinator of the United States private-sector voluntary standardization system. ANSI facilitates the development of American National Standards (ANS) by accrediting the procedures of standards-developing organizations. The Institute remains a private, nonprofit membership organization supported by a diverse constituency of private and public sector organizations. (Source: http://web.ansi.org/)</td></tr>
<tr><td>American Standard Code for Information Interchange</td><td>ASCII</td><td>ASCII is a character set and a character encoding based on the Roman alphabet as used in modern English. ASCII codes represent text in computers, in other communications equipment, and in control devices that work with text. Most often, nowadays, character encoding has an ASCII-like base.

ASCII defines the following printable characters, presented here in numerical order of their ASCII value:

!"#$%'()*+,-./0123456789:; ?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~(

(Source: http://en.wikipedia.org/wiki/ASCII)</td></tr>
<tr><td>Application</td><td></td><td>An application is a software program that performs a specific function directly for a user, with or without requiring extraordinary authority or privileges such as system-level control and monitoring, administrative or "super user" rights, or root-level access. (Source: derived from Committee on National Security Systems Instruction 4009, National Information Assurance Glossary [R1339])</td></tr>
<tr><td>Application Environment Profile</td><td>AEP</td><td>The AEP describes the exact functionality supported by the Operating Environment of the SCA specification.</td></tr>
</table>

Application Programming Interface	API	A special type of interface that specifies the calling conventions with which one component may access the resources and services provided by another component. APIs are defined by sets of procedures or function-invocation specifications. An API is a special case of an interface.
Application Server		A platform for developing and deploying multi-tier distributed enterprise applications.
Architectural Style		An architectural style is the combination of distinctive features in which **architecture** is performed or expressed. (Source: http://www.opengroup.org/projects/soa/doc.tpl?gdid=10632)
Architecture		(1) The structure of components, their relationships, and the principles and guidelines governing their design and evolution over time. (2) A high-level design that provides decisions about the problem(s) that the product will solve, component descriptions, relationships between components, and dynamic operation description. (3) A framework or structure that portrays relationships among all the elements of the subject force, system, or activity. Also, the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. The organizational structure of a system or component, their relationships, and the principles and guidelines governing their design and evolution over time. (Source: IEEE Std 610.12)
Assistant Secretary of Defense for Networks and Information Integration	ASD (NII)	(Source: http://www.dod.mil/nii/)
Asymmetric Key Cryptography		Synonym for **Public Key Cryptography**.
Attribute		A distinct characteristic of an object. Real-world object attributes are often specified in terms of their physical traits, such as size, shape, weight, and color. Cyberspace object attributes might describe size, type of encoding, and network address. (Source: ***Web Services for Remote Portlets Specification***, ***Appendix A: Glossary***; http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf)
Authentication		The process that verifies the identity of a user, device, or other entity in a computer system, usually as a prerequisite to allowing access to resources in a system. The Java servlet specification requires three types of authentication (basic, form-based, and mutual) and supports digest authentication. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Authorization		The process by which access to a method or resource is determined. Authorization depends on the determination of whether the principal associated with a request through authentication is in a given security role. A security role is a logical grouping of users defined by the person

		who assembles the application. A deployer maps security roles to security identities. Security identities may be principals or groups in the operational environment. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Basic Object Adapter	BOA	The Basic Object Adapter was an early (v1) CORBA component; see the **Portable Object Adapter** (**POA**).
Binary XML		Binary XML is a format which does not conform to the XML specification yet maintains a well-defined, useful [i.e., practical systems may take advantage of this relationship with little additional effort] relationship with XML. (Source: derived from Section 2.1 ***Definition of Binary XML*** in the ***XML Binary Characterization W3C Working Group Note***, 31 March 2005; http://www.w3.org/TR/xbc-characterization/)
Business Logic		The code that implements the functionality of an application. In the Enterprise JavaBeans architecture, this logic is implemented by the methods of an enterprise bean. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Business Process Execution Language	BPEL	A Business Process Execution Language provides a means of assembling a set of discrete services into an end-to-end process flow. For example, the **Organization for the Advancement of Structured Information Standards** (**OASIS**) Web Services Business Process Execution Language (WS-BPEL) Version 2.0 [R1347] defines a model and grammar for describing the behavior of business processes.
Canonicalization		The process of converting data that has more than one possible representation into a "standard" canonical representation. This can be done to compare different representations for equivalence, to count the number of distinct data structure , to improve the efficiency of various algorithms by eliminating repeated calculations, or to make it possible to impose a meaningful sorting order. (Source: http://en.wikipedia.org/wiki/Canonicalization) When referring to XML, the process of converting an XML document to a form that is consistent to all parties. Used when signing documents and interpreting signatures. Any XML document is part of a set of XML documents that are logically equivalent within an application context. Generally, if two documents have the same canonical form, then the two documents are logically equivalent within the given application context. Methods exist for generating a physical representation, the canonical form, of an XML document that accounts for the permissible changes. Note that two documents may have differing canonical forms yet still be equivalent in a given context based on application-specific equivalence rules for which no generalized XML specification could account.
Cascading Style Sheet	CSS	Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents. (Source: http://www.w3.org/Style/CSS/)
Certificate	CERT	A certificate which uses a digital signature to bind together a public key with an identity information such as the name of a person or an

		organization, their address, and so forth. The certificate can be used to verify that a public key belongs to an individual. (Source: http://en.wikipedia.org/wiki/Certificate_%28cryptography%29)
Certificate Authority	CA	A trusted organization which issues digital public key certificates for use by other parties. It is an example of a trusted third party. CAs are characteristic of many public key infrastructure (PKI) schemes. (Source: http://en.wikipedia.org/wiki/Certificate_authority)
Certificate Revocation List	CRL	A list of certificates (more accurately, their serial numbers) which have been revoked, are no longer valid, and should not be relied upon by any system user. (Source: http://en.wikipedia.org/wiki/Certificate_Revocation_List)
Check Constraint		A constraint based on a user-defined condition - generally documented in a database domain - that has to evaluate to true for the contents of a data base column to be valid.
Chief Information Officer	CIO	Job title for a manager responsible for **Information Technology** (IT) within an organization; often reports to the chief executive officer or chief financial officer. For information on the Assistant Secretary of Defense for Networks and Information Integration (ASD/NII)/DoD CIO see DoDD 5144.1 of 2 May 2005. (Source: http://en.wikipedia.org/wiki/Chief_Information_Officer)
Client		A system entity that accesses a Web service. (Source: http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf)
Cohesion		The manner and degree to which the tasks performed by a single software module are related to one another. Types include coincidental, communicational, functional, logical, procedural, sequential, and temporal. Synonym: module strength. Contrast with **coupling**. In a well-designed, highly modular software design, the modules will have high cohesion; that is, each will have a clearly defined set of functions that have a close relationship to each other. This facilitates changes to modules since the changes will affect only the closely-related functions. In contrast, modules that contain multiple, unrelated functions blur the integrity of the software's design since the unrelated functions are bound into a single module, thereby creating dependencies that inhibit the ability to easily make changes. (Source: IEEE Std 610.12-1990)
Collaboration		Portal members can communicate synchronously through chat or messaging, or asynchronously through threaded discussion, blogs, and email digests (forums).
Command and Control	C2	(DoD) The exercise of authority and direction by a properly designated commander over assigned and attached forces in the accomplishment of the mission. Command and control functions are performed through an arrangement of personnel, equipment, communications, facilities, and procedures employed by a commander in planning, directing, coordinating, and controlling forces and operations in the

		accomplishment of the mission. (Source: DoD, ***Department of Defense Dictionary of Military and Associated Terms***, JP 1-02, 12 April 2001 as amended through 17 October 2008)
Commercial Off-The-Shelf	COTS	A term for systems that are manufactured commercially, and may be tailored for specific uses. (Source: http://en.wikipedia.org/wiki/Commercial_off-the-shelf)
Common Access Card	CAC	A DoD-wide smart card used as the identification card for active duty Uniformed Services personnel (to include the Selected Reserve), DoD civilian employees, eligible contractor personnel, and eligible foreign nationals; the primary platform for the Public Key Infrastructure (PKI) authentication token used to access DoD computer networks and systems in the unclassified environment and, where authorized by governing security directives, the classified environment; and the principal card enabling physical access to buildings, facilities, installations, and controlled spaces as described in DoD Directive 8190.3, "Smart Card Technology," 31 August 2002. ***Note:*** *The Defense Manpower Data Center (DMDC) Common Access Card site (http://www.dmdc.osd.mil/smartcard) contains additional information, reports and developer support concerning the DoD CAC implementation.* (Source: DoD Instruction 8520.2, 1 April 2004, [R1206] Enclosure (2) Definitions, page 13)
Common Business Oriented Language	COBOL	COBOL is a third-generation programming language. Its name is an acronym, for COmmon Business Oriented Language, defining its primary domain in business, finance, and administrative systems for companies and governments. (Source: http://en.wikipedia.org/wiki/COBOL)
Common Language Runtime	CLR	CLR, at the very core of the **.NET** Framework, encapsulates all the services used from the operating system by compilers of higher level languages such as Visual Basic .NET, Visual C++ .NET, Visual J# .NET and Visual C# .NET. The higher level languages ultimately are translated into native code that directly accesses the CLR.
Common Object Request Broker Architecture	CORBA	CORBA "wraps" code written in another language into a bundle containing additional information on the capabilities of the code inside, and explaining how to call it. The resulting wrapped objects can then be called from other programs (or CORBA objects) over the network. The CORBA specification defines APIs, communication protocol, and object/service information models to enable heterogeneous applications written in various languages running on various platforms to interoperate. (Source: http://en.wikipedia.org/wiki/CORBA)
Community of Interest	COI	A COI is a collaborative group of users that must exchange information in pursuit of its shared goals, interests, missions, or business processes and therefore must have shared vocabulary for the information it exchanges. (Source: DoDD 8320.02, 2 December 2004, *Data Sharing in a Net-Centric Department of Defense*)

Community of Interest Service		A service that may be offered to the enterprise, but is owned and operated by a **Community of Interest** to provide or support a well-defined set of mission functions and associated information.
Compiler		A computer program that translates programs expressed in a high-order language into their machine language equivalent. (Source: IEEE Std 610.12-1990)
Complex Semi-Structured Data		Complex Semi-Structured Data has partial metadata. It includes data defined in COBOL copybooks and Electronic Data Interchange standards ANSI X.12 and Health Level 7 (HL7). Semi-structured data can be as complex or more so as any Complex Structured data. It can map into or be XML. It may also be missing some metadata or an XSD.
Complex Structured Data		Complex Structured Data has well-defined metadata. It includes data represented in XML documents with deeply hierarchical and recursive structures. Complex data can be represented in a complex data structure or can be mapped into a relational or flat structure with additional metadata provided to represent the complex relationships. Although complex structured data is generically a property of object oriented databases, the Complex Data Structures can be filled from any source.
Complex Unstructured Data		Complex Unstructured Data has little or no metadata. It includes data in binary files, spreadsheets, documents, and print streams.
Component		One of the parts that make up a system. A component may be hardware or software and may be subdivided into other components. Note the terms ***module***, ***component***, and ***unit*** are often used interchangeably or defined to be sub-elements of one another in different ways depending on the context. The relationship of these terms is not yet standardized. (Source: IEEE Std 610.12-1990) ***Note:*** *See **system component** and **software component**.*
Component-Based Software		Mission applications that are architected as components integrated within a component framework.
Conceptual Model		Captures the concepts of the relational database and can help enforce the first three normalization rules.
Condition		A variable of the operational environment or situation in which a unit, system, or individual is expected to operate that may affect performance. A **DDS** Condition is attached to a **WaitSet** and indicates which condition the application is waiting for asynchronously: **`StatusCondition`**, **`ReadCondition`** or **`QueryCondition`**.
Confidentiality		The property that data is not made available to unauthorized individuals, entities, or processes.

Configuration Control Board	CCB	Also Change Control Board. Duties include reviewing change requests, making decisions, and communicating decisions made to affected groups and individuals. Represents the interests of program and project management by ensuring that a structured process is used to consider proposed changes and incorporate them into a specified release of a product.
Container		An entity that provides life-cycle management, security, deployment, and runtime services to J2EE components. Each type of container (EJB, Web, JSP, servlet, applet, and application client) also provides component-specific services. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Core Enterprise Services	CES	Core Enterprise Services (CES) are a small set of **services** provided by the Enterprise Information Environment Mission Area (EIEMA). Some of the CES services will be centrally provided on behalf of the DoD while others might involve local provisioning. For locally provisioned services, EIEMA provides guidance to ensure consistent implementation throughout the DoD. (Source: *DoD Net-Centric Services Strategy*, Section 3.1 [R1313])
Coupling		The manner and degree of interdependence between software modules. Types include common-environment coupling, content coupling, control coupling, data coupling, hybrid coupling, and pathological coupling. Contrast with **cohesion**. In a well-designed, highly modular software design, the coupling between modules will be minimized. This facilitates changing and replacing modules with minimal effect on other modules within the system. (Source: IEEE Std 610.12-1990)
CRL Distribution Point	CDP	The location where the **Certificate Authority** (**CA**) puts the **Certificate Revocation List** (**CRL**) for relying parties to obtain the most current CRL.
Data		Unprocessed information; information without context.
Data Architect		A Data Architect is a job title associated with a person within an organization responsible for making sure the organization's strategic goals are optimized through the use of enterprise data standards. This frequently involves creating and maintaining a centralized registry of **metadata**. Data Architecture includes topics such as metadata management, business semantics, data modeling and metadata workflow management. A Data Architect's job frequently includes the set up a **metadata registry** to allow domain-specific stakeholders to maintain their own **data elements**.
Data Asset		Any entity that is composed of data. For example, a database is a data asset that contains data records (e.g., system or application output files, databases, documents, or Web pages). The term data asset also

		refers to services that provide access to data. For example, a service that returns individual records from a database is considered a data asset since it deals mainly in the function of providing data. Similarly, a Web site that returns data in response to specific queries (e.g., www.defenselink.mil) is considered a data asset. (Source: DoD Net-Centric Data Strategy, 9 May 2003 [R1172])
Data at Rest	DAR	Data at Rest refers to all data in computer storage (e.g., on hard disk drives, CDs/DVDs, floppy disks, thumb drives, PDAs, cell phones, other removable storage media, etc.) while excluding data that is traversing a network (data in transit) or temporarily residing in computer memory to be read or updated (data in use). Source: DoD Policy Memorandum, Encryption of Sensitive Unclassified Data at Rest on Mobile Computing Devices and Removable Storage Media R1330: DoD Memorandum , **Encryption of Sensitive Unclassified Data at Rest on Mobile Computing Devices and Removable Storage Media** Chief Information Officer . [http://iase.disa.mil/policy-guidance/dod-dar-tpm-decree07-03-07.pdf]
Database Data		Data stored in database columns in database tables in a relational database. The set of data records which a relational database is populated. Generally understood to refer to application data and not metadata.
Database Management System	DBMS	A system, usually automated and computerized, for managing any collection of compatible, and ideally normalized, data. (Source: http://en.wikipedia.org/wiki/DBMS)
Data-Centric		An approach for the design and implementation of systems, applications, services or software that emphasis the data rather than the operations. It implies that the data is physically separated from the code and consequently can be maintained independently (loose coupling between code and data).
Data-Centric Publish-Subscribe	DCPS	The Data-Centric Publish-Subscribe is a lower level layer of the **DDS** infrastructure that is targeted towards the efficient delivery of the proper information to the proper recipients.
Data Dictionary		A data dictionary is set of metadata that contains definitions and representations of **data elements**. Within the context of a DBMS, a data dictionary is a read-only set of tables and views. The data dictionary may be considered a database in its own right.
Data Distribution Service for Real-Time Systems	DDS	DDS is a recently-adopted OMG standard that is the first open international middleware standard directly addressing publish-subscribe communications for real-time and embedded systems. DDS introduces a virtual Global Data Space where applications can share information by simply reading and writing data-objects addressed by means of an application-defined name (Topic) and a key. DDS features fine and extensive control of QoS parameters, including reliability,

		bandwidth, delivery deadlines, and resource limits. DDS also supports the construction of local object models on top of the Global Data Space. (Source: OMG Data Distribution Portal, http://portals.omg.org/dds)
Data Element		A data element is an atomic unit of data that has the following: • an identification such as a data element name • a clear data element definition • one or more representation terms • optional enumerated values
Data Element Gallery		The Data Element Gallery is an important component of the Metadata Registry and Clearinghouse. The Data Element Gallery provides its users with access to **data elements** that are commonly used by the Department of Defense such as country codes and U.S. state codes. Users may search the registry, compare data elements, and download an Access database containing the available elements. See the DoD Metadata Registry, http://metadata.dod.mil.
Data Exposure		The steps necessary to set up the metadata infrastructure associated with a net-centric data strategy.
Data Local Reconstruction Layer	DLRL	The Data Local Reconstruction Layer is an optional part of the **DDS** specification that provides a higher level layer allowing for a simpler integration of the DDS into the application layer.
Data Modeling	DM	Modeling is an essential step in understanding the data that will comprise a system. The end products of data modeling can be XML schemas or RDBMS schema definitions. Many COIs create their own data models, such as the Joint Command, Control and Consultation Information Exchange Data Model (JC3IEDM) data model for the **C2** community.
Data Publishing		The steps necessary to make data available within the net-centric data strategy infrastructure.
Data Structure		In computer science, a data structure is a way of storing data in a computer so that it can be used efficiently. Often a carefully chosen data structure will allow a more efficient algorithm to be used. The choice of the data structure often begins from the choice of an abstract data structure. A well-designed data structure allows a variety of critical operations to be performed, using as few resources, both execution time and memory space, as possible. Data structures are implemented using the data types, references and operations on them provided by a programming language. (Source: http://en.wikipedia.org/wiki/Data_structure)
Data Type		A data type is a constraint placed upon the interpretation of data in a type system in computer programming. Common types of data in programming languages include primitive types (such as integers, floating point numbers or characters), tuples, records, algebraic data types, abstract data types, reference types, classes and function types.

		A data type describes representation, interpretation and structure of values manipulated by algorithms or objects stored in computer memory or other storage device. The type system uses data type information to check correctness of computer programs that access or manipulate the data. (Source: http://en.wikipedia.org/wiki/Data_type)
DDS DataReader		The **DDS** **`DataReader`** acts as a typed (i.e., dedicated to only one application data type) accessor to a subscriber. The **`DataReader`** class allows the application to declare the data it wishes to receive (i.e., make a subscription) and access the data received by the attached **Subscriber**.
DDS DataWriter		A **DDS** **`DataWriter`** acts as a typed (i.e., dedicated to only one application data type) accessor to a publisher. The **`DataWriter`** class allows the application to set the value of the data to be published under a given **Topic**.
DDS DomainParticipant		A **DDS** domain participant represents the local membership of the computer process in a **domain**. A domain is a distributed concept that links all the computer processes able to communicate with each other. It represents a communication plane; only the **publishers** and the **subscribers** attached to the same domain may interact. A computer process can run on the behalf of some user or application.
DDS Global Data Space		Underlying any **data-centric** publish subscribe system is a data model. In **DDS**, this model defines the global data space and specifies how **Publishers** and **Subscribers** refer to portions of this space. (See DDS **Domain**)
DDS Listener		A **DDS** **`Listener`** is used to provide a callback for synchronous access. Listeners provide a generic mechanism for the middleware to notify the application of relevant asynchronous events, such as arrival of data corresponding to a subscription, violation of a **QoS** setting, etc. Each **DCPS** entity supports its own specialized kind of listener. **`Listener`** operations are invoked using a middleware-provided thread.
DDS Publication		A **DDS** publication is defined by the association of a **DataWriter** to a **publisher**. This association expresses the intent of the application to publish the data described by the DataWriter in the context provided by the publisher.
DDS Publisher		A **DDS** publisher is an object responsible for data distribution. It may publish data of different data types. The **DataWriter** is the object the application must use to communicate to a publisher the existence and value of data-objects of a given type. When data-object values have been communicated to the publisher through the appropriate **`DataWriter`**, it is the publisher's responsibility to perform the distribution (the publisher will do this according to its own QoS, or the **QoS** attached to the corresponding **`DataWriter`**).
DDS Subscriber		A **DDS** subscriber is an object responsible for receiving published data and making it available (according to the Subscriber's **QoS**) to the receiving application. It may receive and dispatch data of different specified types. To access the received data, the application must use a typed **DataReader** attached to the subscriber.

DDS Subscriber Access API		**DDS** defines two **APIs** that provide subscriber access: **Listeners** and the dual **Condition/WaitSet** infrastructure allow applications to be notified when changes occur in a **DCPS** communication.
DDS Subscription		A **DDS** subscription is defined by the association of a **DataReader** with a **subscriber**. This association expresses the intent of the application to subscribe the data described by the **`DataReader`** in the context provided by the subscriber.
DDS WaitSet		A **DDS** **`WaitSet`** associated with one or several **Condition** objects provides asynchronous data access. **`WaitSets`** and their associated **`Conditions`** provide the means for an application thread to block waiting for the same events that can be received via a **Listener**. Using a **`WaitSet`** the application can handle the event in its own thread instead of the middleware provided thread used for **`Listeners`**.
Defense Information Systems Agency	DISA	Combat support agency responsible for planning, engineering, acquiring, fielding, and supporting global net-centric solutions to serve the needs of the President, Vice President, the Secretary of Defense, and other DoD Components, under all conditions of peace and war. (Source: http://www.disa.mil/main/about/missman.html)
Defense IT Standards Registry	DISR	The DoD IT Standards Registry (DISR) is an online repository (https://disronline.csd.disa.mil/) for a minimal set of primarily commercial IT standards formerly captured in the Joint Technical Architecture (JTA), Version 6.0. These standards are used as the "building codes" for all systems being procured in the Department of Defense. Use of these building codes facilitates interoperability among systems and integration of new systems into the Global Information Grid (GIG). In addition, the DISR provides the capability to build profiles of standards that programs will use to deliver net-centric capabilities. (Source: based on https://acc.dau.mil/CommunityBrowser.aspx?id=334024#7.2.5.2)
Department of Defense	DoD	The Department of Defense is America's oldest and largest government agency. The DoD mission is to provide the military forces needed to deter war and to protect the security of the United States. (Source: adapted from *DoD 101, An Introductory Overview of the Department of Defense*; http://www.defenselink.mil/pubs/dod101/; accessed 30 April 2009)
Deployment		The process whereby software is installed into an operational environment. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Deployment Descriptor		An XML file provided with each module and J2EE application that describes how they should be deployed. The deployment descriptor directs a deployment tool to deploy a module or application with specific container options and describes specific configuration requirements that a deployer must resolve. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)

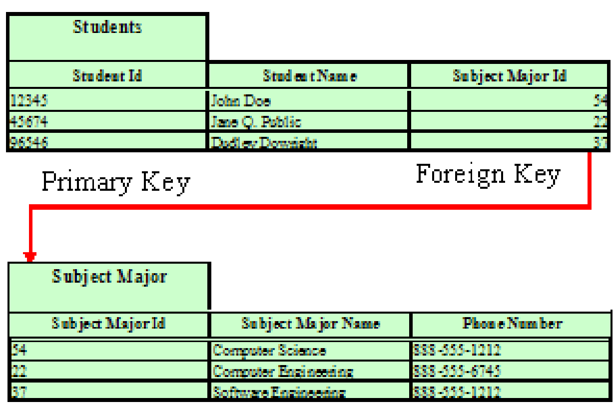
Deprecate		Deprecation is the gradual phasing-out of features such as guidance, software or programming language features. Guidance, features or methods marked as deprecated are considered obsolete, and further use is discouraged. The guidance features or methods are still valid although error messages as warnings may occur when they are referenced. These serve to alert the user to the fact that the feature may be removed in future releases. Features get marked as deprecated, rather than simply removed, in order to provide backward compatibility end users.
Deserialization		Deserialization is the reverse process of **serialization**. A stream of data is converted back into a complex object. ***Note:*** *The process of transferring data using serialization and deserialization is called **marshalling**.*
Design Pattern		General repeatable solution to a commonly-occurring problem in software design. A design pattern isn't a finished design that can be transformed directly into code; it is a description or template for how to solve a problem that can be used in many different situations. (Source: http://en.wikipedia.org/wiki/Design_pattern_%28computer_science%29)
Digest		A cryptographic checksum of an octet stream.
Digital Signature		A value computed with a cryptographic algorithm and bound to data in such a way that intended recipients of the data can use the signature to verify that the data has not been altered and/or has originated from the signer of the message, providing message integrity and authentication. The signature can be computed and verified with symmetric key algorithms, where the same key is used for signing and verifying, or with asymmetric key algorithms, where different keys are used for signing and verifying (a private and public key pair are used).
Digital Signature Algorithm	DSA	The ***Digital Signature Algorithm (DSA)*** is a United States Federal Government standard for digital signatures. It was proposed by the **National Institute of Standards and Technology** (**NIST**) in August 1991 for use in their Digital Signature Standard (DSS), specified in **Federal Information Processing Standard** (**FIPS**) 186, adopted in 1993. A minor revision was issued in 1996 as FIPS 186-1. The standard was expanded further in 2000 as FIPS 186-2 and again in 2009 as FIPS 186-3. (Source: http://en.wikipedia.org/wiki/Digital_Signature_Algorithm; accessed 7 September 2010)
Directory Service		A directory service organizes computerized content and runs on a directory server computer. It is not to be confused with the directory itself, which is the database that holds the information about objects that are to be managed by the directory service. The directory service is the interface to the directory and provides access to the data that is contained in that directory. It acts as a central authority that can securely authenticate resources and manage identities and relationships between them. (Source: http://en.wikipedia.org/wiki/Directory_service)

Discretionary Access Control	DAC	Means of restricting access to objects based on the identity and need-to-know of users and/or groups to which the object belongs. Controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (directly or indirectly) to any other subject. (Source: National Information Assurance (IA) Glossary, CNSSI 4009, revised June 2006)
Distributed Component Object Model	DCOM	Distributed Component Object Model (DCOM) is a Microsoft proprietary technology for software components distributed across several networked computers to communicate with each other. It extends Microsoft's COM, and provides the communication substrate under Microsoft's COM+ application server infrastructure. It has been deprecated in favor of Microsoft **.NET**.
Document Object Model	DOM	The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to access and update the content, structure and style of documents dynamically. (Source: http://www.w3.org/DOM/)
Document Type Definition	DTD	The XML document type declaration contains or points to markup declarations that provide a grammar for a class of documents. This grammar is known as a document type definition, or DTD. The document type declaration can point to an external subset (a special kind of external entity) containing markup declarations, or can contain the markup declarations directly in an internal subset, or can do both. The DTD for a document consists of both subsets taken together. (Source: W3C *Extensible Markup Language (XML) 1.0*, Fifth Edition [R1121])
DoD Discovery Metadata Specification	DDMS	The DoD Discovery Metadata Specification (DDMS) defines discovery metadata elements for resources posted to community and organizational shared spaces. (Source: http://metadata.dod.mil/mdr/irs/DDMS/)
DoD Metadata Registry		As part of the overall **DoD Net-Centric Data Strategy**, the DoD CIO established the DoD Metadata Registry (http://metadata.ces.mil) and a related metadata registration process for the collection, storage and dissemination of structural metadata information resources (schemas, data elements, attributes, document type definitions, style-sheets, data structures, etc.). This Web-based repository is designed to also act as a clearinghouse through which industry and government coordination on metadata technology and related metadata issues can be advanced. As OASD's Executive Agent, **DISA** maintains and operates the ***DoD Metadata Registry and Clearinghouse*** under the direction and oversight of **OASD(NII)**. (Source: DoD Data Services Environment Web site Metadata Registry tab, https://metadata.ces.mil/mdr/about.htm)
DoD Net-Centric Data Strategy	NCDS	This Strategy lays the foundation for realizing the benefits of net-centricity by identifying data goals and approaches for achieving those goals. To realize the vision for net-centric data, two primary objectives must be emphasized: (1) increasing the data that is available to communities or the Enterprise and (2) ensuring that data is usable by both anticipated and unanticipated users and applications. (Source: *Department of Defense Net-Centric Data Strategy*, DoD CIO, 9

		May 2003, http://www.defenselink.mil/cio-nii/docs/Net-Centric-Data-Strategy-2003-05-092.pdf)
DoD PKI Class 3 Assurance Level		Applications handling unclassified medium value information in Moderately Protected Environments, unclassified high value information in Highly Protected Environments, and discretionary access control of classified information in Highly Protected Environments. This assurance level is appropriate for applications that require identification of an entity as a legal person, rather than merely as a member of an organization. ***Note:*** *This definition is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*
DoD PKI High Assurance		Applications that handle high value unclassified information (mission critical) in minimally protected environments require High Assurance certificates. Applications that are applicable for High Assurance certificates include the following: • All applications appropriate for DoD PKI Medium Assurance certificates • Digital signature services for unclassified Mission Assurance Category I (MAC I) or national security information in an unencrypted network • Protection (authentication and confidentiality) for information crossing classification boundaries when such a crossing is already permitted under a system security policy (e.g., sending unclassified information through a High Assurance Guard from **SIPRNet** to **NIPRNet**) (Source: adapted from ***X.509 Certificate Policy for the United States Department of Defense***, Version 9.0, 9 February 2005; http://iase.disa.mil/pki/dod-cp-v90-final-9-feb-05-signed.pdf; DoD PKI Certificate required)
Domain		A group of related items within a certain area of interest. In **DDS**, a domain is the basic construct used to bind individual **publications** and **subscriptions** together for communication. A distributed application can elect to use single or multiple domains for its **data-centric** communications. Domains isolate communication, promote scalability and segregate different classifications of data. (See **Global Data Space**.)
Domain Analysis		The process of identifying the types of information that the data model uses. A good data model captures descriptive information about each of the types.
Domain Name System	DNS	The Domain Name System stores information about hostnames and domain names in a type of distributed database on networks, such as the Internet. Of the many types of information that can be stored, most importantly it provides a physical location (IP address) for each domain name, and lists the mail exchange servers accepting email for each domain.

		The DNS provides a vital service on the Internet as it allows the transmission of technical information in a user-friendly way. While computers and network hardware work with IP addresses to perform tasks such as addressing and routing, humans generally find it easier to work with hostnames and domain names (such as ***www.example.com***) in URLs and email addresses. The DNS therefore mediates between the needs and preferences of humans and of software.
Electronic Data Interchange	EDI	Standard formats for exchanging business data and documents.
Electronic Data Interchange Personnel Identifier	EDI-PI	A unique number assigned to each recipient of a **Common Access Card** (CAC), which is issued by the United States **Department of Defense** through the Defense Enrollment Eligibility Reporting System (DEERS). (Source: http://en.wikipedia.org/wiki/Electronic_Data_Interchange_Personal_Identifier)
Encryption		Encryption is the process of obscuring information to make it unreadable without special knowledge. While encryption has been used to protect communications for centuries, only organizations and individuals with an extraordinary need for secrecy have made use of it. In the mid-1970s, strong encryption emerged from the sole preserve of secretive government agencies into the public domain, and is now employed in protecting widely-used systems, such as Internet e-commerce, mobile telephone networks and bank automatic teller machines. (Source: http://en.wikipedia.org/wiki/Encryption)
Endpoint		The URL or location of the Web service on the internet.
End User		A human user of information. This is distinct from those who develop or support the automated systems that provide the information. -OR- A person who uses a device-specific user agent to access a Web site. (Source: http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf)
Enterprise		An organization considered as an entity or system that includes interdependent resources (e.g., people, organizations, and technology) that must coordinate functions and share information in support of a common mission or a set of related missions. In the computer industry, the term is often used to describe any large organization that utilizes computers. An intranet, for example, is a good example of an enterprise computing system. (Source: http://www.webopedia.com/TERM/e/enterprise.html)
Enterprise Application Integration	EAI	Software to effect interface between enterprise software systems. Provides interface at the application layer.
Enterprise JavaBeans	EJB	A server-side component architecture for the development and deployment of object-oriented, distributed, enterprise-level applications. Applications written using the Enterprise JavaBeans architecture are scalable, transactional, and secure. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)

Enterprise Service		A service that provides capabilities to the enterprise. See also **Core Enterprise Service** and **Community of Interest Service**.
Enterprise Service Bus	ESB	An architectural style that provides distributed invocation, mediation, and end-to-end management and security of services and service interactions to support the larger architectural style known as Service Oriented Architecture (SOA) ***Note***: See the Enterprise Service Bus [P1389] in Part 5 for additional information.
Environment Variable		Environment variables are a set of dynamic values that can affect the way running processes will behave. (Source: http://en.wikipedia.org/wiki/Environment_variable)
eXtensible Access Control Markup Language	XACML	XACML is used to represent and evaluate access control policies. XACML is designed to standardize the use of declarative policy to control access to resources. Used with **SAML**.
eXtensible Markup Language	XML	A markup language defines tags (markup) to identify the content, data, and text in XML documents. It differs from **HTML**, the markup language most often used to present information on the Internet. HTML has fixed tags that deal mainly with style or presentation. An XML document must undergo a transformation into a language with style tags under the control of a style sheet before it can be presented by a browser or other presentation mechanism. Two types of style sheets used with XML are CSS and XSL. Typically, XML is transformed into HTML for presentation. Although tags can be defined as needed in the generation of an XML document, you can use a **document type definition** (**DTD**) to define the elements allowed in a particular type of document. A document can be compared by using the rules in the DTD to determine its validity and to locate particular elements in the document. A Web services application's J2EE deployment descriptors are expressed in XML with schemas defining allowed elements. Programs for processing XML documents use SAX or **DOM** APIs. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
eXtensible Stylesheet Language	XSL	Extensible Stylesheet Language (XSL) is a family of recommendations for defining XML document transformation and presentation. It consists of three parts: • XSL Transformations (XSLT): a language for transforming XML • XML Path Language (XPath): an expression language used by XSLT to access or refer to parts of an XML document • XSL Formatting Objects (XSL-FO): an XML vocabulary for specifying formatting semantics (Source: http://www.w3.org/Style/XSL/)
Facade		Provides a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use. This can simplify a number of complicated object interactions into a single interface.

<table>
<tr><td>Federal Information Processing Standard</td><td>FIPS</td><td>Under the Information Technology Management Reform Act (Public Law 104-106), the Secretary of Commerce approves standards and guidelines that are developed by the National Institute of Standards and Technology (NIST) for Federal computer systems. These standards and guidelines are issued by NIST as Federal Information Processing Standards (FIPS) for use government-wide. NIST develops FIPS when there are compelling Federal government requirements such as for security and interoperability and there are no acceptable industry standards or solutions. (Source: http://www.itl.nist.gov/fipspubs/geninfo.htm)</td></tr>
<tr><td>Font Size</td><td></td><td>The font size refers to the size of the font from baseline to baseline, when set solid (in CSS terms, this is when the font-size and line-height properties have the same value). (Source: http://www.w3.org/TR/REC-CSS2/fonts.html)</td></tr>
<tr><td>FORCEnet</td><td>Fn</td><td>An operational construct and architectural framework that integrates the SEAPOWER21 concepts of Sea Strike, Sea Shield, and Sea Basing by connecting warriors; sensors, networks; command and control; platforms and weapons; providing accelerated speed and accuracy of decision; and integrating knowledge to dominate the battlespace. FORCEnet provides the following capabilities: expeditionary, multi-tiered, sensor and weapon grids; distributed, collaborative, command and control; dynamic, multi-path survivable networks; adaptive/automated decision aids; and human-centric integration.</td></tr>
<tr><td>Foreign Key</td><td>FK</td><td>An attribute in a relation of a database that serves as the primary key of another relation in the same database.

I1156</td></tr>
<tr><td>Global Command and Control System</td><td>GCCS</td><td>GCCS-J is the DOD joint C2 system of record for achieving full spectrum dominance. It enhances information superiority and supports the operational concepts of full-dimensional protection and precision engagement. GCCS-J is the principal foundation for dominant battlespace awareness, providing an integrated, near real-time picture of the battlespace necessary to conduct joint and multinational operations. It fuses select C2 capabilities into a comprehensive, interoperable system by exchanging imagery, intelligence, status of forces, and</td></tr>
</table>

		planning information. GCCS-J offers vital connectivity to the systems the joint warfighter uses to plan, execute, and manage military operations. GCCS-J is a Command, Control, Communications, Computer, and Intelligence (C4I) system, consisting of hardware, software, procedures, standards, and interfaces that provide a robust, seamless C2 capability. The system uses the Defense Information Systems Network (DISN) and must work over tactical communication systems to ensure connectivity with deployed forces in the tactical environment. (Source: http://www.disa.mil/gccs-j/) ***Note:*** *Other variants include GCCS-M to support Maritime operations.*
Hard Code		To hard code or hard coding (also, hard-code/hard-coding, hardcode/hardcoding) refers to the software development practice of embedding output or configuration data directly into the source code of a program or other executable object, or fixed formatting of the data, instead of obtaining that data from external sources or generating data or formatting in the program itself with the given input. Considered an ***anti-pattern*** or ***Bad Thing***, hard coding requires the program's source code to be changed any time the input data or desired format changes, when it might be more convenient to the end user to change the detail by some means outside the program. (Source: http://en.wikipedia.org/wiki/Hard_code; 12 June 2007)
Hierarchical Database		A hierarchical database defines a set of parent-child relationships. Their use should be limited to integration of existing databases, such as IBM's Informational Management System (IMS). Hierarchical database systems require developers to predict all possible access patterns in advance and design the database accordingly. A database access pattern that is not included in the design becomes very difficult and inefficient.
High Availability		Data tier availability can be affected by hardware failure, power outages, data errors, user errors, programmer errors, OS errors, and RDBMS errors. Various hardware and software methods help mitigate availability issues. The more reliable a system needs to be, the more it costs. Consequently, defining availability to meet requirements is essential to controlling costs.
Hypertext Markup Language	HTML	A computer language for representing the contents of a page of hypertext; the language in which most Web pages are currently written. (Source: **W3C** *Glossary*; http://www.w3.org/2003/glossary/alpha/H/)
Hypertext Transfer Protocol	HTTP	A computer protocol for transferring information across the Net in such a way as to meet the demands of a global hypertext system. Part of the original design of the Web, continued in a W3C activity, and now a HTTP 1.1 IETF draft standard. (Source: **W3C** *Glossary*; http://www.w3.org/2003/glossary/alpha/H/)

Hypertext Transmission Protocol Over SSL	HTTPS	HTTPS is the secure version of **HTTP**, the communication protocol of the World Wide Web. It was invented by Netscape Communications Corporation to provide authentication and encrypted communication and is used in electronic commerce. Instead of using plain text socket communication, HTTPS encrypts the session data using either a version of the **SSL** (**Secure Sockets Layer**) protocol or the **TLS** (**Transport Layer Security**) protocol, thus ensuring reasonable protection from eavesdroppers, and man in the middle attacks. The default TCP/IP port of HTTPS is 443. (Source: http://en.wikipedia.org/wiki/HTTPS)
Identification	ID	An act or process that presents an identifier to a system so that the system can recognize a system entity (e.g., user, process, or device) and distinguish that entity from all others. (Source: Committee on National Security Systems Instruction (CNSSI) No. 4009, National Information Assurance (IA) Glossary, 26 April 2010; http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf)
Identity		The set of attribute values (i.e., characteristics) by which an entity is recognizable and that, within the scope of an identity manager's responsibility, is sufficient to distinguish that entity from any other entity. (Source: Committee on National Security Systems Instruction (CNSSI) No. 4009, National Information Assurance (IA) Glossary, 26 April 2010; http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf)
Image Map		An image or graphic that has been coded to contain interactive areas. When it is clicked on, it launches another Web page or program. An image map usually has many different hyperlinked areas, known as links. For example, an image map of a country could be coded so that when a user clicks on a city or region, the browser is routed to a document or Web page about that place. (Source: http://www.netlingo.com/right.cfm?term=clickable%20graphic%20or%20imagemap)
Information		Data to which meaning is assigned, according to context and assumed conventions. Data that has been interpreted, translated, or transformed to reveal the underlying meaning.
Information Assurance	IA	Measures that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation. This includes providing for restoration of information systems by incorporating protection, detection, and reaction capabilities. (Source: DoD Directive 8500.1, *Information Assurance (IA)*, http://www.dtic.mil/whs/directives/corres/pdf/850001p.pdf)
Information Technology	IT	Any equipment or interconnected system or subsystem of equipment, that is used in the automatic acquisition, storage, manipulation, management, movement, control, display, switching, interchange, transmission, or reception of data or information. Information technology includes computers, ancillary equipment, software, firmware, and similar procedures, services (including support services), and related resources. Information technology does not include any equipment that is acquired

		by a federal contractor incidental to a federal contract. (Source: CJCSI 6212.01E, [R1175] Glossary page GL-14)
Integrated Development Environment	IDE	
Integration		Integration is the action or process of combining elements so that they become a whole. Vertical integration acts within a system, whereas horizontal integration acts between or among systems. In the net-centric environment, integration creates links between computer systems, applications, services, or processes. The word is normally used in the context of computing, but can apply to business processes as much as to the underlying process automation. In the past, computer integration such as **enterprise application integration** (EAI) has typically been tightly coupled, or "hard wired," making it difficult to adapt to changing requirements. Thanks to the advent of Web services and the evolution of service-oriented architectures, more agile, loosely coupled forms of integration are starting to emerge.
Integrity		The property whereby an entity has not been modified in an unauthorized manner. (Source: CNSS Instruction No. 4009, 26 April 2010, *National Information Assurance (IA) Glossary* [R1339])
Interface		The functional and physical characteristics required to exist at a common boundary or connection between systems or items. (Source: *Defense Standardization Program (DSP) Policies and Procedures*, DoD 4120.24-M, March 2000) A Key Interface is a common boundary shared between system modules that provides access to critical data, information, materiel, or services; and/or is of high interest due to rapid technological change, a high rate of failure, or costliness of connected modules. (Source: *A Modular Open Systems Approach (MOSA) to Acquisition*, Version 2.0, September 2004; http://www.acq.osd.mil/osjtf/mosapart.html)
Interface Definition Language	IDL	A language used to define interfaces to remote **CORBA** objects. The interfaces are independent of operating systems and programming languages. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Internet		The Internet, or simply the Net, is the publicly available worldwide system of interconnected computer networks that transmit data by packet switching using a standardized Internet Protocol (IP) and many other protocols. It is made up of thousands of smaller commercial, academic, and government networks. It carries various information and services, such as electronic mail, online chat and the interlinked web pages and other documents of the World Wide Web. Because this is by far the largest, most extensive internet (with a lower case i) in the world, it is simply called the Internet (with a capital I). (Source: http://en.wikipedia.org/wiki/Internet)
Internet Engineering Task Force	IETF	The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers

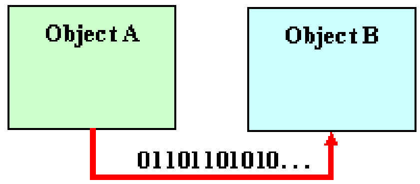
		concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual. (Source: http://www.ietf.org/overview.html)
Internet Inter-ORB Protocol	IIOP	A protocol used for communication between CORBA object request brokers. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Internet Protocol	IP	Data packets routed across network, not switched via dedicated circuits.
Internet Protocol Version 4	IPv4	Version 4 of the Internet Protocol (IP). It was the first version of the Internet Protocol to be widely deployed, and forms the basis for most of the current Internet (as of 2004). It is described in IETF RFC 791, which was first published in September, 1981. IPv4 uses 32-bit addresses, limiting it to 4,294,967,296 unique addresses, many of which are reserved for special purposes such as local networks or **multicast** addresses. This reduces the number of addresses that can be allocated as public Internet addresses. As the number of addresses available is consumed, an IPv4 address shortage appears to be inevitable in the long run. This limitation has helped stimulate the push towards IPv6, which is currently in the early stages of deployment, and may eventually replace IPv4. (Source: http://en.wikipedia.org/wiki/IPv4)
Internet Protocol Version 6	IPv6	Version 6 of the Internet Protocol; it was initially called IP Next Generation (IPng) when it was picked as the winner in the IETF's IPng selection process. IPv6 is intended to replace the previous standard, IPv4, which only supports up to about 4 billion (4×10^9) addresses. IPv6 supports up to about 3.4×10^{38} (340 undecillion) addresses. This is the equivalent of 4.3×10^{20} (430 quintillion) addresses per square inch (6.7×10^{17} (670 quadrillion) addresses/mm^2)of the Earth's surface. It is expected that IPv4 will be supported until at least 2025, to allow time for bugs and system errors to be corrected. (Source: http://en.wikipedia.org/wiki/Ipv6)
Interoperability		The ability of systems, units, or forces to provide data, information, materiel, and services to and accept the same from other systems, units, or forces, and to use the data, information, materiel, and services so exchanged to enable them to operate effectively together. **IT** and **NSS** interoperability includes both the technical exchange of information and the end-to-end operational effectiveness of that exchanged information as required for mission accomplishment. Interoperability is more than just information exchange. It includes systems, processes, procedures, organizations, and missions over the life cycle and must be balanced with **IA**. (Source: CJCSI 6212.01E, *Interoperability and Supportability of Information Technology and National Security Systems*, 15 December 2008)
Intranet		An intranet is a local area network (LAN) used internally in an organization to facilitate communication and access to information that is sometimes access-restricted. Sometimes the term refers only to the most visible service, the internal web site. The same concepts and technologies of the Internet such as clients and servers running on the Internet protocol suite are used to build an intranet. HTTP and other internet protocols are commonly used as well, especially FTP and

		email. There is often an attempt to use internet technologies to provide new interfaces with corporate "legacy" data and information systems. (Source: http://en.wikipedia.org/wiki/Intranet)
ISO/IEC 11179		ISO-11179 (formally known as the ISO/IEC 11179 Metadata Registry (MDR) Standard) is the international standard for representing **metadata** for an organization in a **Metadata Registry**. (Source: http://en.wikipedia.org/wiki/ISO/IEC_11179)
Java		Java is a reflective, object-oriented programming language developed initially by at Sun Microsystems. It was intended to replace C++, although the feature set better resembles that of Objective-C. Java should not be confused with JavaScript, which shares only the name and a similar C-like syntax. Sun Microsystems currently maintains and updates Java regularly. Specifications of the Java language, the Java Virtual Machine (JVM) and the Java API are community-maintained through the Sun-managed Java Community Process.
Java 2 Platform, Enterprise Edition	J2EE	The J2EE environment is the standard for developing component-based multi-tier enterprise applications. The J2EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multitiered, Web-based applications. Features include Web services support and development tools. Sun Microsystems simplified the name of the Java platform for the enterprise, dropping the "2" from the name (**Java Platform, Enterprise Edition**). (Source: based on J2EE 1.4 Glossary, http://www.oracle.com/technetwork/java/javaee/documentation/index.html#120354)
Java Class Files		Class files contain bytecodes for the Java Virtual Machine. They are normally produced by a compiler for the Java programming language. A Java interpreter can then read these files and execute the code contained within.
Java Database Connection	JDBC	An API that supports database and data-source access from Java applications.
Java Development Kit	JDK	The Java Development Kit (JDK) is a superset of the Java Runtime Environment (JRE) and contains everything that is in the JRE plus tools such as the compilers and debuggers necessary for developing applets and applications. The JRE provides the libraries, the Java Virtual Machine, and other components to run applets and applications written in the Java programming language. (Source: http://download.oracle.com/javase/6/docs/)
Javadoc		Javadoc is a tool for generating **API** documentation in **HTML** format from Java source code. The Javadoc tool is included as part of the **Java Development Kit** (**JDK**). (Source: http://www.oracle.com/technetwork/java/javase/documentation/javadoc-137458.html)

Java Message Service	JMS	An API for invoking operations on enterprise messaging systems. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Java Naming and Directory Interface	JNDI	An API that provides naming and directory functionality. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Java Platform, Enterprise Edition	Java EE	Java Platform, Enterprise Edition (Java EE) is an industry standard for enterprise computing. ***Note:*** *Formerly, Java EE was known as* ***Java 2 Platform, Enterprise Edition*** *(**J2EE**).* (Source: http://www.oracle.com/technetwork/java/javaee/overview/index.html)
JavaScript		The Netscape-developed object scripting language used in millions of web pages and server applications worldwide. Contrary to popular misconception, JavaScript is not "Interpretive Java." Rather, it is a dynamic scripting language that supports prototype-based object construction.
JavaServer Pages	JSP	An extensible Web technology that uses static data, JSP elements, and server-side Java objects to generate dynamic content for a client. Typically the static data is HTML or XML elements, and in many cases the client is a Web browser. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Joint Interoperability Test Command	JITC	JITC provides a full-range of agile and cost-effective test, evaluation, and certification services to support rapid acquisition and fielding of global net-centric warfighting capabilities. (Source: http://jitc.fhu.disa.mil/mission.html)
Joint Tactical Radio System	JTRS	JTRS is a family of interoperable, affordable software defined radios which provide secure, wireless networking communications capabilities for Joint forces. (Source: JTRS JPEO, http://jpeojtrs.mil/)
JScript		JScript is the Microsoft implementation of the ECMA-262 language specification (ECMAScript Edition 3). With only a few minor exceptions (to maintain backwards compatibility), JScript is a full implementation of the Ecma International standard. (Source: http://msdn.microsoft.com/en-us/library/14cd3459.aspx)
Just-In-Time Compilation	JIT	This is the primary method by which **.NET** executes **MSIL**. As the MSIL is executed, the code is compiled and optimized for the executing environment. JIT compilation provides environment optimization, runtime type safety, and assembly verification. To accomplish this, the JIT compiler examines the assembly metadata for any illegal accesses and handles violations appropriately.

Key Recovery Manager	KRM	A service of the DOD PKI where copies of key pairs used for encryption are stored and can be recovered for law enforcement purposes. ***Note:*** *This definition is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*
Keystore		A file containing the keys and certificates used for authentication. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Knowledge		(Unlike information or data) Requires the presence of context, semantics, and purpose.
Light Directory Access Protocol	LDAP	The Lightweight Directory Access Protocol (LDAP) is an **Internet** protocol for accessing distributed directory services that act in accordance with X.500 data and service models. (Source: **Internet Engineering Task Force** Request for Comments 4510, ***Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map***, http://tools.ietf.org/html/rfc4510)
Linked Style Sheets		Style sheets that are placed in a separate text files and saved in the root with a css file extension. A link to the file is made in the head section of the document. <head><Break/> <link<Break/> rel="stylesheet"<Break/> href="mystyle.css"<Break/> type="text/css"><Break/></head><Break/>
Look and Feel		Look and feel refers to design aspects of a graphical user interface in terms of colors, shapes, layout, typefaces, etc. (the "look"); and, the behavior of dynamic elements such as buttons, boxes, and menus (the "feel"). It is used in reference to both software and **Web sites**. (Source: http://en.wikipedia.org/wiki/Look_and_feel)
Loosely Coupled		A computing model where application elements require a simple level of coordination and allow for flexible reconfiguration. Interconnection is often asynchronous and message-based.
Lower Camel Case	LCC	A method of naming objects in programming languages which • removes all white space and punctuation between words of the name • uses lower case letters except for the first letter of the second and subsequent words which are upper cased. For example: **`point of contact`** becomes: **`pointOfContact`** ***Note:*** *Also see* ***Upper Camel Case*** *(****UCC****)*

<table>
<tr>
<td>Marshalling</td>
<td></td>
<td>The process of transferring data using serialization and deserialization is called marshalling.

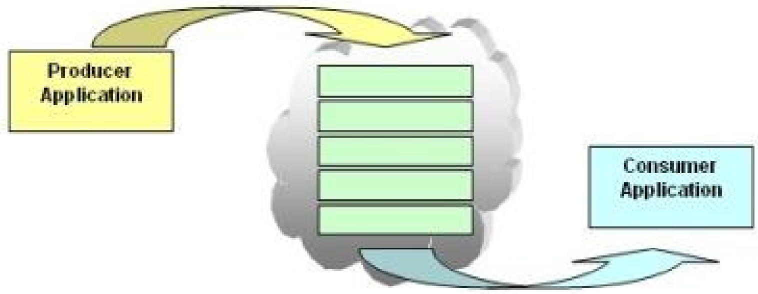


I1158</td>
</tr>
<tr>
<td>Mediation</td>
<td></td>
<td>A set of negotiated agreements for interacting between components that enable those components to work together to perform a task. These agreements are defined through common interfaces and data interchange specifications.

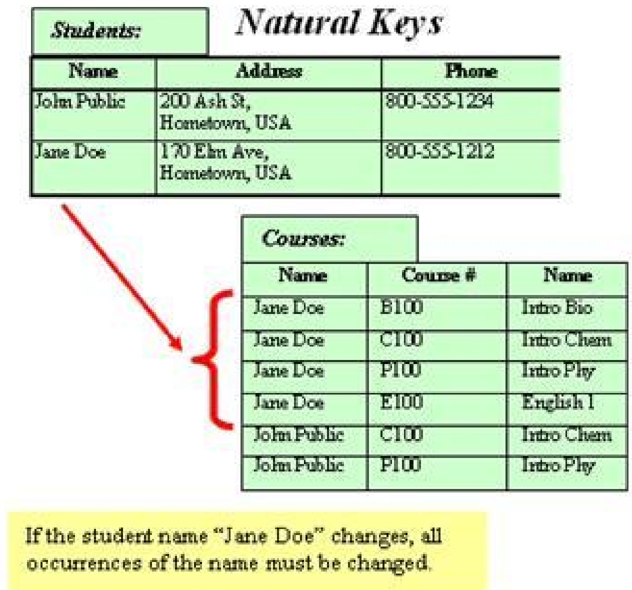
Mediation services provide multiple methods for integrating data sources and services:
<table>
<tr><td>Transformation</td><td>When a client requests data from a service in a particular format, a transformer retrieves and reformats the data before returning it to the client</td></tr>
<tr><td>Aggregation</td><td>A mediator service may collect data derived from multiple sources, thus making many services appear to be one</td></tr>
<tr><td>Adaptation</td><td>When a client cannot communicate directly with a service, an adapter provides service mediation (can be transport protocol as well as data format) when services need to communicate point-to-point</td></tr>
<tr><td>Orchestration</td><td>Co-ordination of events in a process; orchestration directs and manages the on-demand assembly of multiple component services to create a composite application or business process</td></tr>
<tr><td>Choreography</td><td>When a client request spawns a chain of events or service requests that do not rely on a central coordinator, a Choreographed Web Service knows when to execute other services and with which other services to interact; WS-CDL is an example of a business process management workflow language that implements choreography</td></tr>
</table>
</td>
</tr>
<tr>
<td>Message</td>
<td></td>
<td>A self-contained unit of information exchanged between a producer and one or more consumers.

Software commonly uses messages to communicate synchronously or asynchronously between service producers and consumers. Some examples of software messaging are SOAP messages, e-mail messages, Data Distribution Service (DDS) messages, and Java Message Service (JMS) messages.</td>
</tr>
</table>

Message-Oriented Middleware	MOM	Message-oriented middleware acts as an arbitrator between incoming and outgoing messages to insulate producers and consumers from other producers and consumers.
Metadata		Data about the data, that is, the description of the data resources, its characteristics, location, usage, and so on. Metadata is used to identify, describe, and define user data.
Metadata Registry		A Metadata Registry is a central place where metadata definitions are stored and maintained. A metadata registry typically has the following characteristics: • It is a protected area where only approved individuals may make changes • It stores data elements that include both semantics and representations • The semantic areas of a metadata registry contain the meaning of a Data Element with precise definitions • The representational areas define how the data is represented in a specific format such as within a database or a structure file format such as XML Metadata Registries often are stored in an international format called **ISO-11179**.
Microsoft Intermediate Language	MSIL	An intermediate instruction set into which all **.NET** languages compile. You can execute MSIL code on any environment that supports the .NET framework. MSIL-compiled code is verified for safety during runtime, providing better security and reliability than natively compiled binaries. During compilation, .NET code is translated into Microsoft Intermediate Language (MSIL) rather than machine-specific binary code. MSIL is a machine- and platform-independent instruction set that can be executed in any environment within the .NET framework. .NET uses **just-in-time (JIT) compilation** as its primary means of executing MSIL. You can generate native binary images using Microsoft's Native Image Generator (**NGEN**).
Microsoft Message Queue	MSMQ	Messaging in .NET uses Microsoft Message Queue (MSMQ). MSMQ is responsible for reliably delivering messages between applications inside and outside the enterprise. MSMQ ensures reliable delivery by placing messages that fail to reach their intended destination in a queue and then resending them once the destination is reachable.

		 l1067 MSMQ also supports transactions. It permits multiple operations on multiple queues, with all of the operations wrapped in a single transaction, thus ensuring that either all or none of the operations will take effect. Microsoft Distributed Transaction Coordinator (MSDTC) supports transactional access to MSMQ and other resources.
Model-Driven Architecture	MDA	Model-driven architecture is a trademarked term denoting a specific approach to the development of software using models as the basis. The MDA specifies system functionality separately from the implementation of that functionality on a specific technology platform. To accomplish this goal, the MDA defines an architecture that provides a set of guidelines for structuring specifications expressed as models. The MDA model architecture relates multiple standards, including Unified Modeling Language (UML), the Meta Object Facility (MOF), the XML Metadata interchange (XMI), and the Common Warehouse Metamodel (CWM). Note that the term "architecture" in MM does not refer to the architecture of the system being modeled, but rather to the architecture of the various standards and model forms that serve as the technology basis for MDA .
Module		(1) A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to, or output from, an assembler, compiler, linkage editor, or executive routine. (2) A logically separable part of a program. Note: The terms ***module***, ***component***, and ***unit*** are often used interchangeably or defined to be sub-elements of one another in different ways depending upon the context. The relationship of these terms is not yet standardized. See also **component**. (Source: IEEE Std 610.12-1990)
Multicast		The delivery of information to a group of destinations simultaneously using the most efficient strategy to deliver the messages over each link of the network only once and only create copies when the links to the destinations split. (Source: http://en.wikipedia.org/wiki/Multicast)
Namespace		A namespace is an abstract container which contains a logical grouping of unique identifiers (i.e., names). An identifier defined in a namespace is associated with that namespace. It is possible to define the same identifier independently in multiple namespaces. That is, the meaning associated with an identifier defined in one namespace may or may not have the same meaning as the same identifier defined in another namespace. Languages that support namespaces specify the rules that determine to which namespace an identifier (i.e., not its definition) belongs. (Adapted from: http://en.wikipedia.org/wiki/Namespace_%28computer_science%29; accessed 2/6/2008)

		XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references. (Source http://www.w3.org/TR/REC-xml-names/)
National Institute of Standards and Technology	NIST	Non-regulatory federal agency within the U.S. Commerce Department's Technology Administration with a mission to promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life. (Source: http://www.nist.gov/public_affairs/general2.htm)
National Security Agency	NSA	America's cryptologic organization; it coordinates, directs, and performs highly specialized activities to protect U.S. government information systems and produce foreign signals intelligence information. (Source: http://www.nsa.gov/about/index.cfm)
National Security Systems	NSS	Telecommunications and information systems, operated by the Department of Defense, the functions, operation, or use of which involves: (1) intelligence activities; (2) cryptologic activities related to national security; (3) the command and control of military forces; (4) equipment that is an integral part of a weapon or weapons systems; or (5) is critical to the direct fulfillment of military or intelligence missions. Subsection (5) in the preceding sentence does not include procurement of automatic data processing equipment or services to be used for routine administrative and business applications (including payroll, finance, logistics, and personnel management applications). (Source: CJCSI 3170.01F, 1 May 2007, page GL-16)
Native Image Generator	NGEN	NGEN compilation enables you to production of a native binary image of **MSIL** code for the current environment. This improves the performance of the **.NET** application by eliminating the **JIT** overhead associated with the execution. Running NGEN against an assembly, the resulting native image is placed in the Global Assembly Cache for use by all other .NET assemblies. NGEN is a good tool for improving performance of .NET applications as long as the executing environment remains static. If executing an NGEN-generated image in an incompatible environment, .NET automatically reverts to using JIT. To mitigate this, run NGEN during deployment against the installed assemblies.
Native XML Database		Defines a logical model for an XML document (as opposed to the data in that document) and stores and retrieves documents according to that model. These databases are accessed via programming interfaces such as SAX, **DOM**, or JDOM. There is a trend away from pure XML storage because all the leading relational database vendors are introducing advanced XML capabilities.
Natural Key		A Natural Key is a primary keys that is made up completely or in part from naturally occurring data in the tables.

		 I1163 See **Surrogate Key** and **Primary Key**.
Network Centric Warfare	NCW	NCW is an information superiority-enabled concept of operations that generates increased combat power by networking sensors, decision makers, and shooters to achieve shared awareness, increased speed of command, higher tempo of operations, greater lethality, increased survivability, and a degree of selfsynchronization. In essence, NCW translates information superiority into combat power by effectively linking knowledgeable entities in the battlespace. (Source: *Network Centric Warfare: Developing and Leveraging Information Superiority*. David S. Alberts, John J. Garstka and Frederick P. Stien. DoD Command and Control Research Program Publication Series, available at http://www.dodccrp.org/files/Alberts_NCW.pdf)
Node		In general network usage, a node is a processing location such as a computer or some other device. Every node has a unique network address, sometimes called a Data Link Control (DLC) address or Media Access Control (MAC) address. (Source: http://www.webopedia.com/TERM/n/node.html) A NESI Node is a collection of integrated components (i.e., systems, applications, services and other Nodes) that are bound together spatially and/or temporally to meet the needs of a particular mission. It is conceptual in nature and can not be defined in terms of a concrete set of components or size. The membership of a component within a particular Node is not exclusive and a Component can be part of multiple Nodes.
Nonce		A unique random string.
Normalization		Normalization avoids duplication of data, insert anomalies, delete anomalies, and update anomalies. A relation is in first normal form (1NF) if and only if all underlying simple domains contain atomic values only. A relation is in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key. A relation

		is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key.
Object Management Group	OMG	OMG is an international, open membership, not-for-profit computer industry consortium. OMG Task Forces develop enterprise integration standards for a wide range of technologies, and an even wider range of industries. OMG's modeling standards enable powerful visual design, execution and maintenance of software and other processes. OMG's middleware standards and profiles are based on the **Common Object Request Broker Architecture** (CORBA) and support a wide variety of industries. (Source: http://www.omg.org/)
Object-Oriented Analysis	OOA	OOA (Object Oriented Analysis) constitutes the development of software engineering requirements and specifications for a system. These are expressed as an object model (object oriented design) which is composed of a population of interacting objects.
Object-Oriented Databases	OODBMS	Object-oriented databases are based on the object model, and use the same conceptual models as **object-oriented analysis** and **design**.
Object-Oriented Design		Any design that incorporates objects, classes, and inheritance. Contrast with object-based design and class-based design.
Object-Oriented Programming Language		A programming language that enables programmers to define and use objects, classes, and inheritance; for example, C++, Ada 95.
Object Request Broker	ORB	A library that enables **CORBA** objects to locate and communicate with one another. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Online Certificate Status Protocol	OCSP	Online Certificate Status Protocol is a method for determining the revocation status of an X.509 digital certificate using means other than **CRLs**. It is described in RFC 2560 and is on the Internet standards track. OCSP messages are encoded in ASN.1 and usually communicated over **HTTP**. OCSP's request/response nature leads to OCSP servers being termed as OCSP responders.
Online Status Check	OSC	OSC is a service that may be provided by the **Certificate Authority** (**CA**). A relying party sends a request to the OSC service with a certificate, the OSC service responds with a digitally signed response that includes the date and time, certificate identification, and the status of the certificate about whose validity the relying party inquired. The possible responses include "unknown" which may be the response to a query regarding an expired certificate. ***Note:*** *This definition is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Online Status Check Responder	OSCR	OSCR is the server that responds to a relying party's OSC request.
Ontology		An explicit specification of how to represent the objects and concepts that exist in some area of interest and of the relationships that pertain among them. (Source: DoD 8320.02-G, 12 April 2006, Guidance for Implementing Net-Centric Data Sharing)
Open Database Connectivity	ODBC	In computing, Open Database Connectivity (ODBC) provides a software API method for using database management systems (DBMS). The designers of ODBC aimed to make it independent of programming languages, database systems, and operating systems. (Source: adapted from Wikipedia ***Open Database Connectivitiy***, http://en.wikipedia.org/wiki/Odbc; accessed 13 September 2010)
Open Standard		Open standards are publicly available specifications for achieving a specific task. By allowing anyone to obtain and implement the standard, they can increase compatibility between various hardware and software components, since anyone with the necessary technical know-how and resources can build products that work together with those of the other vendors that base their designs on the standard (although patent holders may impose "reasonable and non-discriminatory" royalty fees and other licensing terms on implementers of the standard). Source: http://en.wikipedia.org/wiki/Open_standard) ***Note:*** *NESI restricts the use of the term "standard" to technologies approved by formalized committees that are open to participation by all interested parties and operate on a consensus basis.*
Organization for the Advancement of Structured Information Standards	OASIS	A not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards. (Source: http://www.oasis-open.org/who/)
OS File Systems		A file system that stores and retrieves data, acting as a data tier. Advocates cite performance and simplicity, but the loss of DBMS-inherent capabilities such as ad-hoc queries and the ability to upgrade to faster machines is a deterrent. File-system-based data tiers often result in proprietary solutions that are hard to maintain and port.
Parser		A module that reads in XML data from an input source and breaks it into chunks so that your program knows when it is working with a tag, an attribute, or element data. A non-validating parser ensures that the XML data is well formed but does not verify that it is valid. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Personalization		The ability for portal members to subscribe to specific types of content and services. Users can customize the look and feel of their environment.

Personally Identifiable Information	PII	Personally Identifiable Information is any information about an individual maintained by an agency, including, but not limited to, education, financial transactions, medical history, and criminal or employment history and information which can be used to distinguish or trace an individual's identity. such as their name, social security number, data and place of birth, mother's maiden name, biometric records, etc., including any other personal information which is linked or linkable to an individual. Source: Department of Defense Guidance on Protecting Personally Identifiable Information (PII) R1332: DoD Memorandum , **Department of Defense Guidance on Protecting Personally Identifiable Information (PII)** . [http://iase.disa.mil/policy-guidance/pii-signed-memo-08182006.pdf]
Physical Model		Translates the conceptual model to a particular RDBMS implementation.
Portability		The ease with which a system or component can be transferred from hardware or software environment to another. (Source: IEEE Std 610.12-1990) The level of software portability of any specific product depends on two factors: the design of the product itself, and the characteristics of the source and target execution environments. Software products are rarely if ever 100% portable. Generally, the level of portability depends on the target platform. Software that is highly portable to one class of platform might be not portable to other classes.
Portable Object Adapter	POA	The **Common Object Request Broker Architecture** (**CORBA**) Portable Object Adapter (POA) allows programmers to construct object implementations that are portable across different CORBA **Object Request Broker** (**ORB**) products with minimal changes and recompilation. POAs are specified using the **Interface Definition Language** (**IDL**). (Source: adapted from the ***Common Object Request Broker Architecture (CORBA) Specification***, Version 3.1, Part 1: ***CORBA Interfaces***, http://www.omg.org/spec/CORBA/3.1/Interfaces/PDF/)
Portable Operating System Interface for Computing Environments	POSIX	
Portal		A Web portal is a **Web site** that provides a starting point, gateway, or portal to other resources on the **Internet** or an intranet. Intranet portals are also known as "enterprise information portals" (EIP). Examples of existing portals are Yahoo, Excite, Lycos, Altavista, Infoseek, and Hotbot. (Source: http://en.wikipedia.org/wiki/web_portal)
Portal Page		A complete document rendered by a portal. (Source: http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf)
Portlet		Portlets are pluggable user interface software components that are managed and displayed in a **Web** portal. Portlets produce fragments of markup code that are aggregated into a portal page. Typically, following

		the desktop metaphor, a portal page is displayed as a collection of non-overlapping portlet windows, where each portlet window displays a portlet. Hence a portlet (or collection of portlets) resembles a Web-based application that is hosted in a portal. Portlets may be implemented using various specifications such as the **Web Services for Remote Portlets** (**WSRP**) protocol or the Java Portlet Specification. (Source: adapted from Wikipedia ***Portlet***, http://en.wikipedia.org/wiki/Portlets, accessed 13 September 2010)
Portlet Container		A portlet container provides a runtime environment for **portlets** implemented according to the portlet **API**. In this environment portlets can be instantiated, used, and finally destroyed. The portlet container is not a standalone container like the **servlet** container; instead it is implemented as a thin layer on top of the servlet container and reuses the functionality provided by the servlet container. (Source: http://portals.apache.org/pluto/)
Portlet Specification	JSR 168	To enable interoperability between **portlets** and **portals**, this specification defines a set of **APIs** for portal computing that address the areas of aggregation, personalization, presentation, and security. (Source: http://www.jcp.org/en/jsr/detail?id=168)
Primary Key	PK	An object that uniquely identifies a row within a table.
Private Key		The private key is one of a pair of keys that are generated as part of asymmetric key cryptography. The private key is kept secret; the **public key** can be shared openly with others.
Protocol		An agreed-upon format for transmitting data between two devices. The protocol determines the type of error checking to be used, data compression method, if any, how the sending device will indicate that it has finished sending a message, and how the receiving device will indicate that it has received a message. (Source: http://www.webopedia.com/TERM/p/protocol.html)
Proxy		A **server** that sits between a client application, such as a **Web browser**, and a real server. It intercepts all requests to the real server to see if it can fulfill the requests itself. If not, it forwards the request to the real server. Proxy servers have two main purposes: improve performance and filter requests. (Source: http://www.webopedia.com/TERM/p/proxy_server.html)
Proxy Pattern		Provides a surrogate or placeholder for another object to control access to it.
Public Key	PK	See **Public Key Cryptography**.
Public Key Certificate		Used in client-certificate authentication to enable the server, and optionally the client, to authenticate each other. The public key

		certificate is the digital equivalent of a passport. It is issued by a trusted organization, called a certificate authority, and provides **identification** for the bearer. (Source: based on *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Public Key Cryptography		Public key cryptography, also known as asymmetric cryptography, is a form of cryptography in which a user has a pair of cryptographic keys - a public key and a private key. The private key is kept secret, while the public key may be widely distributed. The keys are related mathematically, but the private key cannot be practically derived from the public key. A message encrypted with the public key can be decrypted only with the corresponding private key. (Source: http://en.wikipedia.org/wiki/Public_key)
Public Key Enabling	PK-Enabling	The incorporation of the use of certificates for security services such as authentication, confidentiality, data integrity, and nonrepudiation. PK-Enabling involves replacing existing or creating new user authentication systems using certificates instead of other technologies, such as userid and password or **Internet Protocol** filtering; implementing public key technology to digitally sign, in a legally enforceable manner, transactions and documents; or using public key technology, generally in conjunction with standard symmetric encryption technology, to encrypt information at rest and/or in transit. (Source: DoD Instruction 8520.2, *Public Key Infrastructure (PKI) and Public Key (PK) Enabling*, 1 April 2004 [R1206])
Public Key Infrastructure	PKI	The framework and services that provide for the generation, production, distribution, control, accounting and destruction of public key certificates. Components include the personnel, policies, processes, server platforms, software, and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, recover, and revoke public key certificates. (Source: CNSS Instruction No. 4009, 26 April 2010, *National Information Assurance (IA) Glossary* [R1339])
Publish/Subscribe Messaging System		A messaging system in which clients address messages to a specific node in a content hierarchy, called a topic. Publishers and subscribers are generally anonymous and can dynamically publish or subscribe to the content hierarchy. The system takes care of distributing the messages arriving from a node's multiple publishers to its multiple subscribers. Messages are generally not persistent and will only be received by subscribers who are listening at the time the message is sent. A special case known as a "durable subscription" allows subscribers to receive messages sent while the subscribers are not active. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Quality of Service	QoS	Data timeliness, accuracy, completeness, integrity, and ease of use. Refers to the probability of the network meeting a given traffic contract. In many cases is used informally to refer to the probability of a packet passing between two points in the network. (Source: http://en.wikipedia.org/wiki/Quality_of_service) -OR- A defined level of performance that adapts to the environment in which it is operating. QoS may be requested by the user of the information. The level of

		QoS provided is based on the request, the available capabilities of the provider, and the priority of the user.
Real-Time		An operation within a larger dynamic system is called a real-time operation if the combined reaction- and operation-time of a task is shorter than the maximum delay that is allowed, in view of circumstances outside the operation. The task must also occur before the system to be controlled becomes unstable. A real-time operation is not necessarily fast, as slow systems can allow slow real-time operations. This applies for all types of dynamically changing systems. The polar opposite of a real-time operation is a batch job with interactive timesharing falling somewhere in-between the two extremes. (Source: http://en.wikipedia.org/wiki/Real_time)
Real-Time System		A system in which the correctness of system behavior depends on both the logical correctness of the computation and the time at which the result is produced. For a real-time system, the system fails if its timing constraints are not met. "Real time" is not necessarily synonymous with "fast." The latency of the response might not be an issue, and it could be on the order of seconds or minutes. But the bounded latency that is sufficient to solve the problem at hand is guaranteed by the system. "Bounded" means that the response is neither too early nor too late. In real-time systems, early can be as bad as late.
Reference Data Set		A reference data set is a collection of related data that represent a defined entity within a Community of Interest. Examples of reference data sets include country codes, U.S. state codes, and marital status codes. (Soure: **DoD Metadata Registry and Clearinghouse**; https://metadata.ces.mil/mdr/other.htm?page=help)
Referential Integrity		A feature provided by RDBMSs that prevents users or applications from entering inconsistent data. Most RDBMSs have various referential integrity rules that you can apply when you create a relationship between two tables.
Registered Namespace		A namespace that has been registered and approved with a **namespace** registration services. For the DoD, use the **DoD Metadata Registry**.
Relational Database	RDB	A collection of data items organized as a set of formally-described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables.
Relational Database Management System	RDBMS	A database management system (DBMS) that is based on the relational model or that presents the data to the user as relations. A collection of tables, each table consisting of a set of rows and columns, can satisfy this property. RDBMSs also provide relational operators to manipulate the data in tabular form. (Source: http://en.wikipedia.org/wiki/RDBMS)
Relative Font Size		Fonts that display according to the size of the surrounding text. Some designers call them scalable fonts. Instead of displaying a fixed pixel size, a relative font size displays as a percentage of the

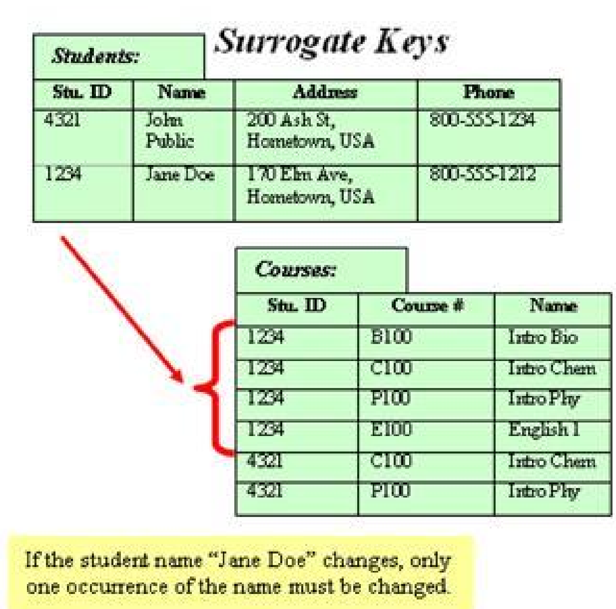
		surrounding elements. (Source: http://www.netmechanic.com/news/vol5/design_no13.htm)
Remote Method Invocation	RMI	A technology that allows an object running in one Java virtual machine to invoke methods on an object running in a different Java virtual machine. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Remote Procedure Call	RPC	An alternative to sockets that abstracts the communication interface to the level of a procedure call. The programmer has the illusion of calling a local procedure, but in fact the arguments of the call are packaged and sent to the remove target of the cell. RPC systems encode arguments and return values using an external data representation such as XDR. RPC does not translate well into distributed object systems, which require communication between program-level objects in different address spaces. To match the semantics of object invocation, distributed object systems require RMI. A local surrogate (stub) object manages the invocation on a remote object.
Representational State Transfer	REST	The Representational State Transfer (REST) architectural style for distributed hypermedia systems was originally defined by Roy Fielding in his Ph.D. dissertation, *Architectural Styles and the Design of Network-based Software Architectures*. One of the authors of the later **HTTP** protocol specifications, he defined a minimalist, stateless-protocol approach to coordinating a service's client and server across a network. RESTful designs adhere to the following constraints: • Client-Server • Stateless • Cacheable • Layered System • Uniform interface Optionally, RESTful designs may also support a sixth constraint: • Code-on-Demand Originally intended for Web hypermedia, the general approach has since been extended to services layered on other protocols and data formats. (Souce: Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000; http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm)
Resource Definition Framework	RDF	
Role-Based Access Control	RBAC	With RBAC, security is managed at a level that corresponds closely to the organization's structure. Each user is assigned one or more roles, and each role is assigned one or more privileges that are permitted to users in that role. Security administration with RBAC consists of determining the operations that must be executed by persons in particular jobs, and assigning employees to the proper roles. Complexities introduced by mutually exclusive roles or role hierarchies are handled by the RBAC software, making security administration

		easier. (Source: National Institute of Standards and Technology Computer Security Resource Center, http://csrc.nist.gov/groups/SNS/rbac/)
Rollback		The point in a transaction when all updates to any resources involved in the transaction are reversed. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Sans Serif Font		A sans serif font is a font that has no serifs. Examples are ***Arial***, ***Century Gothic***, and ***Helvetica***. (Source: http://web.mit.edu/abiword_v2.0.10/Tutorials/klw/glossary.html)
SCA Operating Environment	OE	**SCA** Operating Environment: The SCA OE describes the requirements of the operating system, middleware, and the CF interfaces and operations.
Schema		A diagrammatic representation, an outline, or a model. In relation to data management, a schema can represent any generic model or structure that deals with the organization, format, structure, or relationship of data. Some examples of schemas are (1) a database table and relational structure, (2) a **document type definition** (DTD), (3) a data structure used to pass information between systems, and (4) an **XML schema document** (XSD) that represents a data structure and related information encoded as XML. Schemas typically do not contain information specific to a particular instance of data (Source: DoD 8320.02-G, 12 April 2006, *Guidance for Implementing Net-Centric Data Sharing*)
Secret Internet Protocol Router Network	SIPRNet	SIPRNet is DoD's largest interoperable command and control data network, supporting the **Global Command and Control System** (GCCS), the Defense Message System (DMS), collaborative planning and numerous other classified warfighter applications. Direct connection data rates range from 56 kbps to 155 Mbps. Remote dial-up services are available up to 19.2 kbps. (Source: http://www.disa.mil/services/data.html)
Secret Key		The asymmetric key cryptography approach generates two keys, a public key and a private key. The **private key** is often referred to as the secret key.
Secure Hash Algorithm	SHA	The SHA (Secure Hash Algorithm) family is a set of related cryptographic hash functions. In cryptography, a cryptographic hash function is a hash function with certain additional security properties to make it suitable for use as a primitive in various information security applications, such as authentication and message integrity. A hash function takes a long string (or message) of any length as input and produces a fixed length string as output, sometimes termed a message **digest** or a digital fingerprint. (Source: http://en.wikipedia.org/wiki/SHA#SHA-0_and_SHA-1)
Secure Sockets Layer	SSL	A protocol for transmitting private documents via the Internet. SSL uses a cryptographic system employing two keys to encrypt data: a **public**

		key known to everyone and a **private** or **secret key** known only to the recipient of the message. (Source: http://www.webopedia.com/TERM/S/SSL.html)
Security Assertion Markup Language	SAML	The Security Assertion Markup Language (SAML) is a set of specifications describing security assertions that are encoded in **XML**, profiles for attaching the assertions to various protocols and frameworks, the request/response protocol used to obtain the assertions, and bindings of this protocol to various transfer protocols (for example, **SOAP** and **HTTP**). (Source: *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*, http://docs.oasis-open.org/security/saml/v2.0/saml-glossary-2.0-os.pdf)
Semantics		The implied meaning of data, the study or words and their meanings.
Serialization		Serialization is the process of writing a complex object into a serial stream of data. When the data is successfully transferred, the data can be **deserialized** back into a complex object. ***Note:*** *The process of transferring data using serialization and **deserialization** is called **marshalling**.*
Serif Font		A serif is a feature of the letters in a given typeset. They appear at the end of lines within the letters. An example would be the letter T in Times New Roman - at the end of each horizontal line is a tick that hangs down (that is the serif). Serif fonts include ***Times New Roman***, ***Bookman Oldstyle***, and ***Courier***.  I1165
Server		A computer software application that carries out some task (i.e., provides a service) on behalf of yet another piece of software called a **client**.
Service		A service is an autonomous encapsulation of some business or mission functionality. The service concept includes the notion of service providers and service consumers interacting via well-defined reusable interfaces. ***Note:*** *See the Service-Oriented Architecture [P1304] perspective in Part 1 for additional information concerning services including implementation characteristics.*
Service Level Agreement	SLA	A contractual vehicle between a service provider and a service consumer. It specifies performance requirements, measures of

		effectiveness, reporting, cost, and recourse. It usually defines repair turnaround times for users.
Service-Oriented Architecture	SOA	NESI describes SOA as an architectural style used to design, develop, and deploy information technology (IT) systems based on decomposing functionality into services with well-defined interfaces. ***Note:*** *See the Service-Oriented Architecture [P1304] perspective in Part 1 for additional information.*
Service Provider		The person, organization, or automated asset that implements and operates a service.
Service Registry		Provides descriptive information about a service, enabling the lookup and discovery of services.
Servlet		A Java program that extends the functionality of a Web server, generating dynamic content and interacting with Web applications using a request-response paradigm. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Session		An interaction between system entities of finite duration, often involving a user, typified by the maintenance of some state of the interaction for the duration of the interaction. (Source: http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf)
Session Key		A session key is an encryption and decryption key randomly generated to ensure the security of a communications session between a user and a computer or between two computers. Session keys are sometimes called symmetric keys, because the same key is used for both encryption and decryption. Throughout each session, the key is transmitted with each message and is encrypted with the recipient's public key. Because much of their security relies upon the brevity of their use, session keys are often changed frequently.
Simple Mail Transfer Protocol	SMTP	The objective of the Simple Mail Transfer Protocol (SMTP) is to transfer mail reliably and efficiently. SMTP is independent of the particular transmission subsystem and requires only a reliable ordered data stream channel. (Source: Internet Engineering Task Force (IETF) Request for Comments (RFC) 5321 available at http://tools.ietf.org/html/rfc5321)
Simple Structured Data		Simple Structured Data has an uncomplicated data structure. All requisite **metadata** is provided and simple data types only are used (e.g., integers, long integers, strings, and simple lists.
Simple Unstructured Data		Simple Unstructured Data has uncomplicated data structure but not all requisite **metadata** is provided.

Single Sign-On	SSO	
Single Touch Point		The portal becomes the delivery mechanism for all business information services.
Smart Card		A credit card-size device, normally for carrying and use by personnel, that contains one or more integrated circuits and also may employ one or more of the following technologies: magnetic stripe, bar codes (linear and two-dimensional), non-contact and radio frequency transmitters, biometric information, encryption and authentication, or photo identification. (Source: DoD Directive 8190.3, *Smart Card Technology*, 31 August 2003, Page 2, Section 3.2)
SOAP		SOAP Version 1.2 is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics. (Source: SOAP Version 1.2 Second Edition, http://www.w3.org/TR/soap12-part1/#intro) ***Note:*** *The World Wide Web Consortium (W3C) changed the name of this protocol from* ***Simple Object Access Protocol 1.1 (SOAP)*** *to* ***SOAP Version 1.2*** *in the current version.*
Software Communications Architecture	SCA	An implementation-independent framework for the development of software for an established hardware platform, such as software defined radios.
Software Component		A software component is a software system element offering a predefined service and able to communicate with other components. It is a unit of independent deployment and versioning, encapsulated, multiple-use, non-context-specific and composeable with other components. (Source: http://en.wikipedia.org/wiki/Software_component#Software_component)
Stored Procedure		A unit or module of code that executes in a database and implement some bit of application logic or business rule. Often written in proprietary language such as Oracle's PL/SQL or Sybase's Transact-SQL.
Stovepipe System		A stovepipe system is a legacy system that is an assemblage of inter-related elements that are so tightly bound together that the individual elements cannot be differentiated, upgraded or refactored. The stovepipe system must be maintained until it can be entirely replaced by a new system. Examples of stovepipe systems: • Systems for which new hardware is no longer available • Systems whose original source code has been lost • Systems that were built using old or ad hoc engineering methodologies for which support can no longer be found

		The term is also used to describe a system that does not interoperate with other systems, presuming instead that it is the only extant system. A stovepipe system is an example of an anti-pattern legacy system and demonstrates software brittleness. (Source: http://en.wikipedia.org/wiki/Stovepipe_system)
Structured Query Language	SQL	The standardized relational database language for defining database objects and manipulating data. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Structured Query Language 1992	SQL-92	The SQL-92 and SQL:1999 standards are very detailed and specific. At the current time, no **RDBMS** vendors fully support the entire standard. Vendors that claim they are SQL-92-compliant or SQL:1999-compliant are actually only compliant to a certain level. The SQL-92 standard defines the following levels, which also apply to SQL:1999: (1) Notational; (2) Transitional level SQL92; (3) Intermediate level SQL92; (4) .Full SQL92. (Source:http://dbs.uni-leipzig.de/en/lokal/standards.pdf; http://developer.mimer.com/documentation/html_82/Mimer_SQL_Reference_Manual/Intro_SQL_Stds3.html)
Structured Query Language 1999	SQL-99	See **SQL-92**.
Style Sheet		Style sheets describe how documents are presented on screens, in print, or perhaps how they are pronounced. (Source: http://www.w3.org/Style)
Surrogate Key		A surrogate key is a primary key that has been explicitly created and has no relationship with the naturally occurring data found within a table.  I1167 See **Natural Key** and **Primary Key**.

Symmetric Key Algorithm		Encryption algorithm where the same key is used for both encrypting and decrypting a message.
System		A system is a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce systems-level results. The results include system level qualities, properties, characteristics, functions, behavior and performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are interconnected (Rechtin, 2000). (Source: International Council on Systems Enginering, *A consensus of the INCOSE Fellows*, http://www.incose.org/practice/fellowsconsensus.aspx)
System Component		A basic part of a system. System components may be personnel, hardware, software, facilities, data, material, services, and/or techniques that satisfy one or more requirements in the lowest levels of the functional architecture. System components may be subsystems and/or configuration items. ***Note:*** *See* ***component****.*
Taxonomy		The science of categorization, or classification, of things based on a predetermined system. In reference to Web sites and portals, a site's taxonomy is the way it organizes its data into categories and subcategories, sometimes displayed in a site map. (Source: http://www.webopedia.com/TERM/t/taxonomy.html)
Taxonomy Gallery		The Taxonomy Gallery [of the **DoD Metadata Registry and Clearinghouse**] provides XML-based **taxonomy** files that describe one or more nodes in a hierarchical classification of items, and their relationships to other nodes. The taxonomy files registered with the Taxonomy Gallery are organized by governance namespace. (Source: http://www.disa.mil/nces/development/developer_doc_overview.html; URL no longer accessible, 27 July 2011)
Tenet		Net-centric design precept.
Topic		Topics are used to manage content flow between publishers and subscribers. Topics must be known in such a way that subscribers can refer to them unambiguously. In **DDS**, Topics conceptually fits between **publications** and **subscriptions** and associate a name (unique in the **domain**), a data-type, and **QoS** parameters related to the data.
Transaction		A set of input data that triggers execution of a specific processor job. Usually manipulates data that may need to be rolled back to the original values if any part of the transaction fails. Transactions enable multiple users to access the same data concurrently. (Source: based on *J2EE*

		1.4 Glossary, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Transmission Control Protocol	TCP	One of the core protocols of the Internet protocol suite. Using TCP, programs on networked computers can create connections to one another, over which they can send data. The protocol guarantees that data sent by one endpoint will be received in the same order by the other, without any pieces missing. It also distinguishes data for different applications (such as a Web server and an email server) on the same computer. (Source: http://en.wikipedia.org/wiki/Transmission_Control_Protocol)
Transmission Control Protocol/Internet Protocol	TCP/IP	TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications. The TCP provides for reliable inter-process communication between pairs of processes in host computers attached to distinct but interconnected computer communication networks. (Source: **Internet Engineering Task Force** Request for Comments 793, ***Transmission Control Protocol: DARPA Internet Program Protocol***, September 1981, http://tools.ietf.org/rfc/rfc0793.txt)
Transport Layer Security	TLS	A protocol that guarantees privacy and data integrity between client/server applications communicating over the Internet. The TLS protocol is made up of two layers: • The TLS Record Protocol -- layered on top of a reliable transport protocol, such as TCP, it ensures that the connection is private by using symmetric data encryption and it ensures that the connection is reliable. The TLS Record Protocol also is used for encapsulation of higher-level protocols, such as the TLS Handshake Protocol. • The TLS Handshake Protocol -- allows authentication between the server and client and the negotiation of an encryption algorithm and cryptographic keys before the application protocol transmits or receives any data. (Source: http://www.webopedia.com/TERM/T/TLS.html)
Trigger		In a DBMS, a trigger is a SQL procedure that initiates (fires) an action when an event (INSERT, DELETE, or UPDATE) occurs. Since triggers are event-driven specialized procedures, the DBMS stores and manages them. A trigger cannot be called or executed; the DBMS automatically fires the trigger as a result of a data modification to the associated table. Triggers maintain the referential integrity of data by changing the data in a systematic fashion.
Triple Data Encryption Algorithm	TDEA	An encryption algorithm whose key consists of three DES (Data Encryption Standard) keys, which is also referred to as a key bundle. A DES key consists of 64 binary digits ("0"s or "1"s) of which 56 bits are randomly generated and used directly by the algorithm. (The other 8 bits, which are not used by the algorithm, may be used for error detection.) Each TDEA encryption/decryption operation (as specified in ANSI X9.52) is a compound operation of DES encryption and decryption operations. Let EK(I) and DK(I) represent the DES encryption and decryption of

		I using DES key ***K*** respectively. (Source: http://www.atis.org/tg2k/_triple_data_encryption_algorithm.html)
Trusted Platform Module	TPM	The TPM is a microcontroller that stores keys, passwords and digital certificates. It typically is affixed to the motherboard of computers. It potentially can be used in any computing device that requires these functions. The nature of this hardware chip ensures that the information stored there is made more secure from external software attack and physical theft. The TPM standard is a product of the Trusted Computing Group consortium. Source: Encryption of Sensitive Unclassified Data at Test on Mobile Computing Devices and Removable Storage Media R1330: DoD Memorandum , **Encryption of Sensitive Unclassified Data at Rest on Mobile Computing Devices and Removable Storage Media** Chief Information Officer . [http://iase.disa.mil/policy-guidance/dod-dar-tpm-decree07-03-07.pdf]
Trust Point		A trust point is a **Certificate Authority** (**CA**) that is the root of all trust for all CAs in a CA hierarchy.
Tunneling		Transporting IPv6 traffic through IPv4 networks by encapsulating IPv6 packet in IPv4 and vice-versa.
Unclassified but Sensitive Internet Protocol Router Network	NIPRNet	The Unclassified but Sensitive Internet Protocol (IP) Router Network (NIPRNet) is a global long-haul IP based network to support unclassified IP data communications services for combat support applications to the Department of Defense (DoD), Joint Chiefs of Staff (JS), Military Departments (MILDEPS), and Combatant Commands (COCOM). NIPRNet provides seamless interoperability IP services to customers with access data rates ranging from 56KB to 1.0GB via direct connections to a NIPRNet router, remote dial-up services (56KB), services to the Tactical community via ITSDN/STEP sites, and access to the Internet. (Source: http://www.disa.mil/services/data.html)
Unicode		A standard defined by the Unicode Consortium. Unicode uses a 16-bit code page that maps digits to characters in languages around the world. Because 16 bits covers 32,768 codes, Unicode is large enough to include all the world's languages, with the exception of ideographic languages that have a different character for every concept, such as Chinese. For more information, see http://www.unicode.org/. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Unified Class Library		With the introduction of **.NET**, Microsoft redesigned the access to common system components and services such as XML Web services, Enterprise Services, ADO.NET, and XML by creating a single object-oriented library. All the Microsoft Visual .NET languages (Visual Basic, C++, J#, C#, etc.) have access to this library. To make access to these objects available within the various languages, Microsoft provided infrastructure such as hierarchical namespaces, structures, types, and common objects like collections.

Unified Modeling Language	UML	In the field of software engineering, the Unified Modeling Language (UML) is a standardized specification language for object modeling. UML is a general-purpose modeling language that includes a graphical notation used to create an abstract model of a system, referred to as a UML model. UML is officially defined at the **Object Management Group** (**OMG**) by the UML metamodel, a Meta-Object Facility metamodel (MOF). (Source: http://en.wikipedia.org/wiki/Unified_Modeling_Language; 30 March 2007)
Uniform Resource Identifier	URI	An encoded address that represents any Web resource, such as an **HTML** document, image, video clip, or program. As opposed to a **URL** or a **URN**, which are concrete entities, a URI is an abstract superclass. (Source: http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html)
Uniform Resource Locator	URL	A sequence of characters that represents information resources on a computer or in a network such as the Internet. This sequence of characters includes (1) the abbreviated name of the protocol used to access the information resource and (2) the information used by the protocol to locate the information resource.(Source: http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html)
Uniform Resource Name	URN	A name that uniquely identifies a **Web service** to a **client**. (Source: http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html)
UNIQUE Key Integrity Constraint		A **`UNIQUE`** key integrity constraint requires that every value in a column or set of columns (key) be unique; that is, no two rows of a table have duplicate values in a specified column or set of columns. (Source: http://www.lc.leidenuniv.nl/awcourse/oracle/server.920/a96524/c22integ.htm)
Universal Description, Discovery, and Integration	UDDI	An industry initiative to create a platform-independent, open framework for describing services, discovering businesses, and integrating business services using the Internet, as well as a registry. It is being developed by a vendor consortium. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Upper Camel Case	UCC	A method of naming objects in programming languages which • removes all white space and punctuation between words of the name • all letters but the first letter of each word is lower cased. For example: **`point of contact`** becomes: **`PointOfContact`** ***Note:*** *Also see **Lower Camel Case** (LCC).*
Use-Case		A sequence of actions, performed by a system, that yields a result of value to a user. A set of actions, including variants, that a system performs that yields an observable result of value to a particular actor.

User Datagram Protocol	UDP	A connectionless protocol that, like **TCP**, runs on top of **Internet Protocol** (IP) networks. Unlike **Transmission Control Protocol/ Internet Protocol** (TCP/IP), UDP/IP provides very few error recovery services, offering instead a direct way to send and receive datagrams over an IP network. It's used primarily for broadcasting messages over a network. (Source: http://www.webopedia.com/TERM/U/User_Datagram_Protocol.html)
Valid		A valid XML document has data that conforms to a particular set of user-defined content rules, or XML Schemas, that describe correct data values and locations. For example, if an element in a document is required to contain text that can be interpreted as being an integer numeric value, and it instead has the text ***hello***, is empty, or has other elements in its content, then the document is not valid. (Source: adapted from http://en.wikipedia.org/wiki/XML; 9/11/2006)
Very High Speed Integrated Circuit	VHSIC	Specific type of digital logic circuit.
VHDL Component		Special piece of conventional code that allows the construction of hierarchical circuit designs.
VHSIC Hardware Description Language	VHDL	Commonly used design-entry language in the electronic design automation of digital circuits.
Visual Basic Scripting	VBScript	VBScript (Visual Basic Scripting) is a programming language developed by Microsoft which is similar to **JavaScript**. It is used to embed code into **HTML** pages. It is actually a subset of Microsoft's Visual Basic. (Source: Strategic Web Ventures Glossary, http://www.strategicwebventures.com/definitions/Glossary/VBScript)
VoiceXML	VXML	VoiceXML (VXML) is the **W3C** standard **XML** format for specifying interactive voice dialogues between a human and a computer. It is fully analogous to **HTML**, and brings the same advantages of **Web application** development and deployment to voice applications that HTML brings to visual applications. Just as HTML documents are interpreted by a visual web browser, VoiceXML documents are interpreted by a voice browser. A common architecture is to deploy banks of voice browsers attached to the public switched telephone network (PSTN) so that users can simply pick up a phone to interact with voice applications. VoiceXML has tags that instruct the voice browser to provide speech synthesis, automatic speech recognition, dialog management, and soundfile playback.
Web Application		An application written for the Internet, including those built with Java technologies such as JavaServer Pages and servlets, and those built with non-Java technologies such as CGI and Perl. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Web Browser		A client program that initiates requests to a **Web server** and displays the information that the server returns. (Source: http://

		publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html)
Web Container		A container that implements the Web-component contract of the **J2EE** architecture. This contract specifies a runtime environment for Web components that includes security, concurrency, life-cycle management, transaction, deployment, and other services. A Web container provides the same services as a **JSP** container as well as a federated view of the J2EE platform **APIs**. A Web container is provided by a Web or J2EE server. (Source: *J2EE 1.4 Glossary*, http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Web Ontology Language	OWL	The OWL 2 Web Ontology Language, informally ***OWL 2***, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents. (Source: http://www.w3.org/TR/owl2-overview/)
Web Page		A document created with **HTML** (**Hypertext Markup Language**) that is part of a group of hypertext documents or resources available on the World Wide Web. Collectively, these documents and resources form what is known as a **Web site**. You can read HTML documents that reside somewhere on the Internet or on your local hard drive with software called a **Web browser**. Web pages can contain hypertext links to other places within the same document, to other documents at the same Web site, or to documents at other Web sites.
Web Server		Software that provides services to access the Internet, an intranet, or an extranet. A Web server hosts **Web sites**, provides support for HTTP and other protocols, and executes server-side programs (such as Common Gateway Interface (CGI) scripts or servlets) that perform certain functions. In the **J2EE** architecture, a Web server provides services to a **Web container**. For example, a Web container typically relies on a Web server to provide **HTTP** message handling. The J2EE architecture assumes that a Web container is hosted by a Web server from the same vendor, so it does not specify the contract between these two entities. A Web server can host one or more Web containers. (Source: http://www.oracle.com/technetwork/java/javaee/index-jsp-139417.html)
Web Service		A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format. Web service implementation can use any number of technologies and standards including **SOAP** messages and **REST**. (Source: http://www.w3.org/TR/ws-gloss/)
Web Services Description Language	WSDL	WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. (Source: W3C Note on WSDL 1.1 of 15 March 2001 http://www.w3.org/TR/wsdl)

Web Services for Interactive Applications	WSIA	
Web Services for Remote Portlets	WSRP	The WSRP specification defines a **Web service** interface for interacting with interactive presentation-oriented Web services. It has been produced through the joint efforts of the Web Services for Interactive Applications (**WSIA**) and Web Services for Remote Portals (WSRP) OASIS Technical Committees. Scenarios that motivate WSRP/ WSIA functionality include (1) **portal** servers providing **portlets** as presentation-oriented Web services that can be used by aggregation engines; (2) portal servers consuming presentation-oriented Web services provided by portal or non-portal content providers and integrating them into a portal framework. (Source: http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf)
Web Services Interoperability Organization	WS-I	WS-I is an open industry organization chartered to promote Web services interoperability across platforms, operating systems and programming languages. The organization's diverse community of Web services leaders helps customers to develop interoperable Web services by providing guidance, recommended practices and supporting resources. (Source: http://www.ws-i.org/about/Default.aspx)
Web Site		A Web site, website, or WWW site (often shortened to just "site") is a collection of Web pages (i.e., HTML/XHTML documents accessible via **HTTP** on the Internet). All publicly accessible Web sites in existence comprise the World Wide Web. The pages of a Web site are accessed from a common root URL, the homepage, and usually reside on the same physical server. The URLs of the pages organize them into a hierarchy, although the hyperlinks between them control how the reader perceives the overall structure and how the traffic flows between the different parts of the site. (Source: http://en.wikipedia.org/wiki/web_site)
Well-Formed		A textual object is a well-formed **XML document** if: 1. Taken as a whole, it matches the production labeled document. 2. It meets all the well-formedness constraints given in this specification. 3. Each of the parsed entities which is referenced directly or indirectly within the document is well-formed. (Source: http://www.w3.org/TR/REC-xml/#dt-wellformed)
Wireless Application Protocol	WAP	WAP is an open international standard for applications that use wireless communication, such as Internet access from a mobile phone. WAP provides services equivalent to a Web browser with some mobile-specific additions. It is specifically designed to address the limitations of very small portable devices. During its first years of existence WAP suffered from considerable negative media attention and has been criticised heavily for its design choices and limitations. (Source: http://en.wikipedia.org/wiki/WAP)

Wireless Markup Language	WML	WML is the primary content format for devices that implement the **WAP** (Wireless Application Protocol) specification based on XML, such as mobile phones. (Source: http://en.wikipedia.org/wiki/Wireless_Markup_Language)
Wire Protocol		In a network, it is the mechanism for transmitting data from point a. to point b. It often refers to a distributed object protocol such as **Remote Method Invocation** (**RMI**), which is software only and which invokes the running of programs on remote servers. (Source: http://www.techweb.com/encyclopedia/defineterm.jhtml?term=wire+protocol)
Wisdom		Knowledge with information so thoroughly assimilated as to have produced sagacity, judgment, and insight. The ability to use knowledge for a purpose.
World Wide Web	WWW	The World Wide Web ("WWW," or simply "Web") is an information space in which items of interest, referred to as resources, are identified by global identifiers called **Uniform Resource Identifiers** (**URI**). The term is often mistakenly used as a synonym for the **Internet**, but the web is actually a service that operates over the Internet. (Source: http://en.wikipedia.org/wiki/World_Wide_web)
World Wide Web Consortium	W3C	The World Wide Web Consortium (W3C) is an international consortium where Member organizations, a full-time staff, and the public work together to develop Web standards. W3C's mission is to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web. (Source: http://www.w3.org/Consortium/)
XML Attribute		An XML structural construct. A name-value pair, separated by an equals sign, included inside a tagged element that modifies certain features of the element. All attribute values, including things like size and width, are in fact text strings and not numbers. For XML, all values must be enclosed in quotation marks. Attributes can be declared for an XML element type using an attribute list declaration. (Source: http://msdn2.microsoft.com/en-us/library/ms256452.aspx)
XML Document		A document object that is **well-formed**, according to the XML recommendation, and that might (or might not) be valid. The XML document has a logical structure (composed of declarations, elements, comments, character references, and processing instructions) and a physical structure (composed of entities, starting with the root, or document entity). (Source: http://msdn2.microsoft.com/en-us/library/ms256452.aspx)
XML Element		An XML structural construct. An XML element consists of a start tag, an end tag, and the information between the tags, which is often referred to as the contents. Each element has a type, identified by name, sometimes called its "generic identifier" (GI), and may have a set of attribute specifications. Each attribute specification has a name and a value. An instance of an element is declared using <element> tags. Elements used in an XML file are described by a **DTD** or schema, either

		of which can provide a description of the structure of the data. (Source: http://msdn2.microsoft.com/en-us/library/ms256452.aspx)
XML Gallery		The XML Gallery [of the **DoD Metadata Registry and Clearinghouse**] contains information resources such as submission packages, elements, attributes, and schemas that have been registered by DOD software developers. These information resources use XML, a platform and vendor independent format for exchanging data, to handle data, data structures, and data descriptions (metadata). (Source: http://www.disa.mil/nces/development/developer_doc_overview.html; URL no longer active, 27 July 2011)
XML Information Resources		**Document Type Definition** (**DTD**) or **XML Schema Documents** (**XSD**) files.
XML Instance Document		An XML document defined by an XML Schema but is populated with the data, not the definition of the data.
XML Path Language	XPath	The result of an effort to provide a common syntax and semantics for functionality shared between XSL Transformations (XSLT) and XML Pointer Language (XPointer) . The primary purpose of XPath is to address parts of an XML document. It also provides basic facilities for manipulation of strings, numbers, and Booleans. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values. XPath gets its name from its use of a path notation as used in URLs for navigating through the hierarchical structure of an XML document. (Source: http://msdn2.microsoft.com/en-us/library/ms256452.aspx)
XML Schema		A database-inspired method for specifying constraints on documents using an XML-based language. Schemas address deficiencies in **DTDs**, such as the inability to constrain the kinds of data that can occur in a particular field. Because schemas are founded on XML, they are hierarchical. Thus it is easier to create an unambiguous specification, and it is possible to determine the scope over which a comment is meant to apply.
XML Schema Definition	XSD	A language proposed by the **W3C** XML Schema Working Group for use in defining schemas. Schemas are useful for enforcing structure and/or constraining the types of data that can be used validly within other XML documents. XML Schema Definition refers to the fully specified and currently recommended standard for use in authoring XML schemas. Because the XSD specification was only recently finalized, support for it was only made available with the release of MSXML 4.0. It carries out the same basic tasks as DTD, but with more power and flexibility. Unlike DTD, which requires its own language and syntax, XSD uses XML syntax for its language. XSD closely resembles and extends the capabilities of XDR. Unlike XDR, which was implemented and made available by Microsoft in MSXML 2.0 and later releases, the W3C now recommends the use of XSD as a standard for defining XML schemas. (Source: http://msdn2.microsoft.com/en-us/library/ms256452.aspx)

XSL Transformations	XSLT	A language to express the transformation of XML documents into other XML documents. (Source: W3C Glossary)

References

R1001	World Wide Web Consortium (W3C) , **Document Object Model** . [http://www.w3.org/DOM/]
R1002	Latest **SOAP** Versions, http://www.w3.org/TR/soap/
R1003	**Web Service Definition Language** (**WSDL**) - http://www.w3.org/TR/wsdl
R1008	OASIS , **Web Services Security: SOAP Message Security 1.1 (WS-Security 2004 [commonly known as WS-Security])** . [http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf]
R1023	For answers to frequently asked questions about **cascading style sheets**, see http://www.blooberry.com/indexdot/css/topics/stylefaq.htm
R1024	*Don't Make Me Think* by Steve Krug (ISBN 0-7897-2310-7)
R1026	*Designing Web Usability* by Jakob Nielsen (ISBN 1-56205-810-X)
R1027	OMG - http://www.omg.org/gettingstarted/gettingstartedindex.htm
R1031	Adapter pattern - http://c2.com/cgi/wiki?AdapterPattern
R1032	Design patterns: Proxy - http://www.dofactory.com/Patterns/PatternProxy.aspx
R1033	Facade pattern - http://c2.com/cgi/wiki?FacadePattern
R1034	Java EE - http://www.oracle.com/technetwork/java/javaee/overview/index.html
R1035	EJB - http://java.sun.com/products/ejb/
R1036	.jar - http://java.sun.com/developer/Books/javaprogramming/JAR/
R1037	.war - http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/WCC3.html
R1038	.ear - http://download.oracle.com/javaee/1.4/tutorial/doc/Overview5.html
R1039	.rar - http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Connector2.html
R1041	Model-view-controller - a Whatis.com definition - http://whatis.techtarget.com/definition/0,,sid9_gci214607,00.html
R1042	Oracle Developer Network , **Java Servlet Technology** . [http://www.oracle.com/technetwork/java/javaee/servlet/index.html]
R1046	*The Semantic Web*, Michael C. Daconata, Leo J. Obrst, Kevin T. Smith; Wiley Publishing Inc., 2003
R1047	, **Website Indexing: Enhancing Sccess to Information within Websites** Browne, Glenda and Jeremy, Jon . [http://www.webindexing.biz/joomla/index.php?option=com_content&task=view&id=127&Itemid=119]

R1048	W3C , **OWL 2 Web Ontology Language (W3C Recommendation)** . [http://www.w3.org/TR/owl2-overview/]
R1049	, **Web Services Conversation Language (WSCL) 1.0 (WS Note)** . [http://www.w3.org/TR/wscl10/]
R1052	XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999 [http://www.w3.org/TR/xslt]
R1053	W3C , **XSLT 2.0 (W3C Working Draft)** . [http://www.w3.org/TR/xslt20]
R1055	World Wide Web Consortium (W3C) , **Extensible Stylesheet Language (XSL) Version 1.1** . [http://www.w3.org/TR/xsl/]
R1056	The Extensible Stylesheet Language Family http://www.w3.org/Style/XSL
R1058	CSS (Cascading Style Sheets) versions 1 (CSS1) and 2 (CSS2) [See http://www.w3.org/Style/CSS, http://www.w3.org/TR/REC-CSS1, http://www.w3.org/TR/REC-CSS2]
R1070	Multilateral Interoperability Programme (MIP) , **Joint Command, Control and Consultation Information Exchange Data Model (JC3IEDM)** . [https://mipsite.lsec.dnd.ca/Pages/Documentsoverview.aspx]
R1078	Sun Developer Network (Oracle Corp.) , **Java SE Desktop Overview** . [http://java.sun.com/javase/technologies/desktop/]
R1092	For information on WSRP access specifications, see http://www.ibm.com/developerworks/webservices/library/ws-wsrp/
R1093	For information on JSR-168 access specifications, see http://www.jcp.org/aboutJava/communityprocess/final/jsr168/
R1108	JPEO JTRS , **Software Communication Architecture (SCA)** . [http://www.public.navy.mil/jpeojtrs/sca/Pages/default.aspx]
R1109	Object Management Group (OMG) , **Minimum CORBA Specification, v1.0** . [http://www.omg.org/spec/CORBA/]
R1110	Object Management Group (OMG) , **Lightweight Log Service Specification, v1.1** . [http://www.omg.org/spec/LtLOG/]
R1111	Object Management Group (OMG) , **Software-Based Communication DTF Info Page** . [http://sbc.omg.org/swradio_info.htm#WIP]
R1112	Software Communications Architecture (Wikipedia)
R1113	*Circuit Design with VHDL*, Volnei A. Pedroni, MIT Press, 2004. ISBN: 0-262-16224-5
R1114	*VHDL Coding Styles and Methodologies* (2nd Edition) by Ben Cohen, Kluwer Academic Publishers, 1999. ISBN: 0-7923-8474-1
R1115	*Joint Program Executive Office (JPEO) Joint Tactical Radio System (JTRS) Software Standards*, maintained by the Space and Naval Warfare Systems Center, Pacific, JTRS Standards Team (SSC Pacific Code 55260).

R1116	World Wide Web Consortium (W3C) , **Extensible Markup Language (XML)** . [http://www.w3.org/XML/]
R1117	http://xfront.com/BestPracticesHomepage.html
R1118	Microsoft Developer Network (MSDN) , **Microsoft Standards Reference** . [http://msdn.microsoft.com/en-us/library/ms256177.aspx]
R1119	Component Organization and Registration Environment - https://www.collab.core.gov/CommunityBrowser.aspx?id=2234
R1120	Federal XML Naming and Design Rules - http://xml.coverpages.org/Federal-NDR-20050609.pdf
R1121	W3C , **Extensible Markup Language (XML) 1.0 (Fifth Edition)** . [http://www.w3.org/TR/2008/REC-xml-20081126/]
R1122	W3 Schools XML Syntax Rules Tutorial - http://www.w3schools.com/xml/xml_syntax.asp
R1123	W3 Schools XML Validator Tutorial - http://www.w3schools.com/xml/xml_validator.asp
R1124	W3C , **XML Schema Description Document for the XML Schema Specification** . [http://www.w3.org/2001/XMLSchema.xsd]
R1125	http://www.xfront.com/files/xml-schema.html
R1126	http://www.xfront.com
R1127	http://www.xfront.com/ZeroOneOrManyNamespaces.pdf
R1128	http://www.xfront.com/DefaultNamespace.pdf
R1129	Extensible Content Models - http://www.xfront.com/ExtensibleContentModels.pdf
R1130	Element versus Type - http://www.xfront.com/ElementVersusType.pdf
R1131	XML Schema Part 2: Datatypes Second Edition - http://www.w3.org/TR/xmlschema-2/#built-in-datatypes
R1132	Composition versus Subclassing - http://www.xfront.com/composition-versus-subclassing.html
R1133	World Wide Web Consortium (W3C) , **XML Path Language (XPath)** . [http://www.w3.org/TR/xpath/]
R1134	W3Schools , **XPath Tutorial** . [http://www.w3schools.com/xpath/default.asp]
R1136	HOWTO: Write Namespace-Agnostic XPath and XSLT - http://jcooney.net/post/2005/08/08/HOWTO-Write-Namespace-Agnostic-XPath-and-XSLT.aspx
R1138	, **About SAX** . [http://www.saxproject.org/] This is the official website for SAX (the Simple API for XML); accessed 13 August 2010.
R1140	Human Engineering MIL-STD 1472F, section 5.14 "User Computer Interface"

R1141	"Guide for developing usable and useful web sites" [http://usability.gov]
R1142	"Microsoft Windows User Experience: Official Guidelines for User Interface Developers and Designers," Redmond, WA: Microsoft Press, 1999
R1143	Apple, Inc. , **Apple Human Interface Guidelines** . [http://developer.apple.com/library/mac/#documentation/UserExperience/Conceptual/AppleHIGuidelines/Intro/Intro.html]
R1145	"The GNOME Usability Project. GNOME Human Interface Guidelines (1.0)," [http://developer.gnome.org/projects/gup/hig/1.0/]
R1146	Sun Microsystems , **Java Look and Feel Design Guidelines: Advanced Topics** . [http://java.sun.com/products/jlf/at/book/index.html]
R1147	Sun Microsystems, Inc. , **Java Look and Feel Design Guidelines Second Edition** . [http://java.sun.com/products/jlf/ed2/book/index.html]
R1148	"Designing Interfaces: Patterns for Effective Interaction Design," Tidwell, J., O'Reilly Media, Inc., 2006
R1150	, **C++ Coding Standards, 101 Rules, Guidelines and Best Practices** Herb Sutter and Andrei Alexandrescu .
R1151	"Web-Based Portal Computer-Human Interface Guidelines," Ahlstrom, V. & Allendoerfer, K., 2004 [http://hf.tc.faa.gov/products/bibliographic/tn0423.htm]
R1152	"Web Application Design Handbook: Best Practices for Web-Based Software," Fowler, S. & Stanwick, V., San Francisco: Morgan Kaufmann Publishers, 2004.
R1154	"Federal IT Accessibility Initiative," [http://www.section508.gov/]
R1155	"Electronic and Information Technology Accessibility Standards," Federal Register, [http://www.access-board.gov/sec508/508standards.pdf]
R1156	"Web Content Accessibility Guidelines 1.0," W3C, [http://www.w3.org/TR/WAI-WEBCONTENT/]
R1157	Microsoft Developer Network (MSDN) , **Guidelines for Keyboard User Interface Design** . [http://msdn.microsoft.com/library/?url=/library/en-us/dnacc/html/ATG_KeyboardShortcuts.asp]
R1159	"Internationalization Best Practices: Specifying Language in XHTML & HTML Content," W3C, [http://www.w3.org/TR/i18n-html-tech-lang/]
R1160	"Internationalization Quick Tips for the Web," W3C [http://www.w3.org/International/quicktips/]
R1161	"Developing and Localizing International Software," Madell, T., Parsons, C. & Abegg, J., Englewood Cliffs, NJ: Prentice Hall, 1994
R1162	, **Programming for the World: A Guide to Internationalization** O'Donnell, S.M. (Englewood Cliffs, NJ: Prentice Hall).
R1163	"Software Internationalization and Localization: An Introduction," Uren, E., Howard, R. & Perinotti,T., New York: Van Nostrand Reinhold, 1993

R1164	DoD Directive 5000.01, *The Defense Acquisition System*, 12 May 2003 (certified current as of 20 November 2007); http://www.dtic.mil/whs/directives/corres/pdf/500001p.pdf.
R1165	DoD Instruction 5000.02, *Operation of the Defense Acquisition System*, 8 December 2008; http://www.dtic.mil/whs/directives/corres/pdf/500002p.pdf.
R1167	DoD Directive 4630.05, *Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS)*, 05 May 2004 (certified current as of 23 April 2007); http://www.dtic.mil/whs/directives/corres/pdf/463005p.pdf.
R1168	DoD Instruction 4630.8, *Procedures for Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS)*, 30 June 2004; http://www.dtic.mil/whs/directives/corres/pdf/463008p.pdf.
R1170	DoD CIO , **Global Information Grid Architectural Vision** . [http://cio-nii.defense.gov/docs/gigarchvision.pdf]
R1171	DoD Deputy CIO None , **DoD Architecture Framework (DoDAF)** . [http://cio-nii.defense.gov/sites/dodaf20/]
R1172	*DoD Net-Centric Data Strategy*, DoD Chief Information Officer, 9 May 2003, http://cio-nii.defense.gov/docs/Net-Centric-Data-Strategy-2003-05-092.pdf
R1173	CJCSI 3170.01G, *Joint Capabilities Integration and Development System*, 01 March 2009; http://www.dtic.mil/cjcs_directives/cdata/unlimit/3170_01.pdf.
R1174	CJCSM 3170.01C, *Operation of the Joint Capabilities Integration and Development System*, 01 May 2007; http://www.dtic.mil/cjcs_directives/cdata/unlimit/m317001.pdf.
R1175	CJCSI 6212.01E, *Interoperability and Supportability of Information Technology and National Security Systems*, 15 December 2008; http://www.dtic.mil/cjcs_directives/cdata/unlimit/6212_01.pdf.
R1176	*Net-Centric Operations and Warfare Reference Model (NCOW RM)*, v1.1, 17 November 2005. ***Note:*** *CJCSI 6212.01E removed the NCOW RM element of the Net-Ready Key Performance Parameter (NR-KPP), integrating the components of the former NCOW RM into other elements of the NR-KPP.*
R1177	*Net-Centric Checklist*, V2.1.3, Office of the Assistant Secretary of Defense for Networks and Information Integration/Department of Defense Chief Information Officer, 12 May 2004; http://cio-nii.defense.gov/docs/NetCentric_Checklist_v2-1-3_.pdf.
R1178	*A Modular Open Systems Approach (MOSA) to Acquisition*, Version 2.0, September 2004; http://www.acq.osd.mil/osjtf/mosapart.html.
R1179	, **DoD IT Standards Registry (DISR)** . [https://disronline.csd.disa.mil/] Note: Valid DoD PKI Certificate reqired to access site; non-registered users can view or download the current DISR release (sorted by several different categories) from the Reports & Archives page (see DISR site for link).
R1180	*Net-Centric Attributes List*, Office of the Assistant Secretary of Defense for Networks and Information Integration/Department of Defense Chief Information Officer, 2 February 2007; http://cio-nii.defense.gov/docs/NetCentricAttributesOfficial.pdf .

R1181	*Global Information Grid (GIG) Key Interface Profiles (KIPs) Framework* (DRAFT), Version 0.95, 7 October 2005.
R1197	DoD Directive 8500.01E, *Information Assurance (IA)*, 24 October 2004 (certified current as of 23 April 2007). This directive establishes policy and assigns responsibilities under 10 U.S.C. 2224 to achieve Department of Defense information assurance (IA) through a defense-in-depth approach that integrates the capabilities of personnel, operations, and technology, and supports the evolution to net-centric warfare.
R1198	DoD Instruction 8500.2, **Information Assurance (IA) Implementation** . [http://www.dtic.mil/whs/directives/corres/pdf/850002p.pdf]
R1199	DoD Instruction 8580.01, *Information Assurance (IA) in the Defense Acquisition System* ***This instruction implements policy, assigns responsibilities, and prescribes procedures necessary to integrate Information Assurance (IA) into the Defense Acquisition System; describes required and recommended levels of IA activities relative to the acquisition of systems and services; describes the essential elements of an Acquisition IA Strategy, its applicability, and prescribes an Acquisition IA Strategy submission and review process.***
R1202	Object Management Group (OMG) , **Data Distribution Service for Real-time Systems Specification, v1.2** . [http://www.omg.org/technology/documents/formal/data_distribution.htm]
R1203	**OMG** Data Distribution Portal (http://portals.omg.org/dds)
R1206	DoD Instruction 8520.2; 24 May 2011; *Public Key Infrastructure (PKI) and Public Key (PK) Enabling*; http://www.dtic.mil/whs/directives/corres/pdf/852002p.pdf
R1216	Global Information Grid Net-Centric Implementation Document - Service Definition Framework (S300); Version 2.0; 21 December 2005 (available via Defense Knowledge Online [DKO] at https://www.us.army.mil/suite/page/384284; DKO account and CAC required for access).
R1217	DoD 8320.02-G, 12 April 2006, *Guidance for Implementing Net-Centric Data Sharing*; http://www.dtic.mil/whs/directives/corres/pdf/832002g.pdf
R1222	NIST SP 800-95, "Guide to Secure Web Services" dated August 2007 http://csrc.nist.gov/publications/nistpubs/800-95/SP800-95.pdf
R1223	Web Services Interoperability (WS-I) Profiles: http://www.ws-i.org/deliverables/Default.aspx
R1227	Department of Defense (DoD) , **Metadata Registry** . [https://metadata.ces.mil]
R1232	DoD Directive 5230.09, *Clearance of DoD Information for Public Release*, 22 August 2008
R1237	Web Services Interoperability Organization (WS-I) , **Basic Security Profile, v1.1** . [http://www.ws-i.org/]
R1245	DoD Directive 8320.02, ***Data Sharing in a Net-Centric Department of Defense***, 2 December 2004 (certified current as of 23 April 2007)
R1246	OASIS , **Web Services Security: SAML Token Profile 1 .1** . [http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf]

R1258	Assistant Secretary of Defense for Networks and Information Integration, Memorandum; *Joint Net-Centric Capabilities*, 15 July 2003
R1276	W3C *Namespaces in XML 1.0*, http://www.w3.org/TR/REC-xml-names; *Namespaces in XML 1.1*, http://www.w3.org/TR/xml-names11
R1280	Universal Description, Discovery, and Integration (**UDDI**), http://www.oasis-open.org/committees/uddi-spec
R1289	Javadoc Tool Home Page, http://www.oracle.com/technetwork/java/javase/documentation/javadoc-137458.html.
R1290	Microsoft Developer Network (MSDN) , **XML Documentation Comments (C# Programming Guide)** . [http://msdn.microsoft.com/en-us/library/b2s063f7.aspx]
R1292	DoD Instruction 8552.01, ***Use of Mobile Code Technologies in DoD Information Systems***, 23 October 2006 (available at http://www.dtic.mil/whs/directives/corres/pdf/855201p.pdf)
R1293	Space and Naval Warfare Systems Command (SPAWAR), ***DOD Domain Controller Public Key Infrastructure (DOD-PKI) Domain Controller Administrator Operations Guide (DCAOP)***, 30 May 2006; https://infosec.navy.mil/clt/index.jsp (user registration and DoD PKI Certificate required for access)
R1294	United States Air Force Public Key Infrastructure System Program Office (USAF PKI SPO), ***Configuration and Operations Guide For Air Force Smart Card Certificate-Based Logon Using DoD PKI Domain Controller Certificates***, April 2006; https://afpki.lackland.af.mil/html/sclogon.asp (DoD PKI Certificate required for access)
R1295	Army IA NETCOM, ***Common Access Card (CAC) Cryptographic Logon (CCL) Technical Configuration Guide***, V 1.0, February 2006; https://www.us.army.mil/suite/page/237211; user account (Army or Defense Knowledge Online, AKO or DKO) and DoD PKI Certificate required for access.
R1296	United States Marine Corps , **Cryptographic Logon Enabler (CLOE)** . [http://www.mceits.usmc.mil/docs/cloe_2.pdf] The USMC CLOE document is available via the Marine Corps Enterprise IT Services (MCEITS) Interim Portal Services (iPS), (accessed 9 August 2010).
R1297	DoD Directive 8190.3, ***Smart Card Technology***, 31 August 2002; http://www.dtic.mil/whs/directives/corres/pdf/819003p.pdf
R1298	Carnegie Mellon University Software Engineering Institute CERT, ***Secure Coding Standards***; https://www.securecoding.cert.org/confluence/display/seccode/CERT+Secure+Coding+Standards
R1299	Common Weakness Enumeration; http://cwe.mitre.org/index.html
R1300	Open Web Application Security Project (OWASP), ***Top Ten Most Critical Web Application Security Vulnerabilities***; http://www.owasp.org/index.php/OWASP_Top_Ten_Project
R1301	Carnegie Mellon University Software Engineering Institute Computer Emergency Readiness Team (CERT) , **C++ Secure Coding Standard** . [https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=637]

R1302	Microsoft Developer Network (MSDN) , **Secure Coding Guidelines for the .NET Framework** . [http://msdn.microsoft.com/en-us/library/aa302372.aspx]
R1303	Sun Microsystems, ***Secure Coding Guidelines for the Java Programming Language***; http://www.oracle.com/technetwork/java/seccodeguide-139067.html
R1304	University of Virginia, Department of Computer Science, Inexpensive Program Analysis Group, ***Splint - Secure Programming Lint***; http://lclint.cs.virginia.edu/
R1305	Sun Microsystems, ***Java Annotations***; http://download.oracle.com/javase/tutorial/java/javaOO/annotations.html
R1306	Microsoft Developer Network (MSDN) , **Selective Modification of the Behavior of Compiler Warning Messages** . [http://msdn.microsoft.com/en-us/library/ms879818.aspx]
R1313	*DoD Net-Centric Services Strategy*, DoD CIO, 4 May 2007, http://cio-nii.defense.gov/docs/Services_Strategy.pdf
R1330	DoD Memorandum , **Encryption of Sensitive Unclassified Data at Rest on Mobile Computing Devices and Removable Storage Media** Chief Information Officer . [http://iase.disa.mil/policy-guidance/dod-dar-tpm-decree07-03-07.pdf]
R1331	DoD Directive 8100.02, **Use of Commercial Wireless Devices, Services, and Technologies in the Department of Defense (DoD) Global Information Grid (GIG)** . [http://www.dtic.mil/whs/directives/corres/pdf/810002p.pdf]
R1332	DoD Memorandum , **Department of Defense Guidance on Protecting Personally Identifiable Information (PII)** . [http://iase.disa.mil/policy-guidance/pii-signed-memo-08182006.pdf]
R1333	DoD CIO Memorandum , **Protection of Sensitive DoD Data at Rest on Portable Computing Devices** . [https://www.rmda.army.mil/privacy/docs/foia-DoDSctyGuidPortableComputers.pdf]
R1334	General Services Administration (GSA) Memo 10359, **Data at Rest (DAR) Encryption Awardees Announced** . [http://www.gsa.gov/portal/content/102647]
R1338	Information Assurance Technology Analysis Center (IATIC) , **Software Security Assurance: A State-of-the-Art Report (SOAR)** . [http://iac.dtic.mil/iatac/download/security.pdf]
R1339	Committee on National Security Systems (CNSS) Instruction 4009, **National Information Assurance (IA) Glossary** . [http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf]
R1340	Software Assurance (SwA) Acquisition Working Group None 8510.01, **Software Assurance in Acquisition: Mitigating Risks to the Enterprise** ASD(NII)/CIO . [https://buildsecurityin.us-cert.gov/swa/downloads/SwA_in_Acquisition_102208.pdf]
R1341	DoD Directive 8000.01, **Management of the Department of Defense Information Enterprise** ASD(NII)/DoD CIO . [http://www.dtic.mil/whs/directives/corres/pdf/800001p.pdf]
R1342	DoD CIO None , **Department of Defense Global Information Grid Architectural Vision** . [http://cio-nii.defense.gov/docs/GIGArchVision.pdf]
R1343	DoD None , **Net-Centric Environment Joint Functional Concept** . [http://www.dtic.mil/futurejointwarfare/concepts/netcentric_jfc.pdf]

R1344	DoD Directive 5144.1, **Assistant Secretary of Defense for Networks and Information Integration/DoD Chief Information Officer (ASD(NII)/DoD CIO)** . [http://www.dtic.mil/whs/directives/corres/pdf/514401p.pdf]
R1345	ASD(NII)/DoD CIO None , **Deputy Assistant Secretary of Defense for Cyber, Identity, and Information Assurance Strategy** . [http://cio-nii.defense.gov/docs/DoD_IA_Strategic_Plan.pdf]
R1347	OASIS , **Web Services Business Process Execution Language (WS-BPEL) Version 2.0** . [http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf] OASIS Standard; accessed 6 August 2010 via http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.
R1349	, **Architectural Styles and the Design of Network-based Software Architectures** Fielding, Roy Thomas . [http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm] Note: The URL is for the HTML version of the subject Doctoral dissertation.
R1350	Ecma International , **Technical Report TR/84: Common Language Infrastructure (CLI) - Information Derived from Partition IV XML File (4th Edition)** . [http://www.ecma-international.org/publications/techreports/E-TR-084.htm]
R1351	Ecma International , **Technical Report TR/89: Common Language Infrastructure (CLI) - Common Generics (2nd Edition)** . [http://www.ecma-international.org/publications/techreports/E-TR-089.htm]
R1355	National Institute of Standards and Technology , **Special Publications (800 Series)** . [http://csrc.nist.gov/publications/PubsSPs.html] The NIST Computer Security Resource Center (CSRC) Publications page for Special Publications in the 800 series provides an index of documents of general interest to the computer security community. NIST established the Special Publication 800 series in 1990 to provide a separate identity for information technology security publications.
R1356	National Institute of Standards and Technology None Special Publication 800-92, **Guide to Computer Security Log Management** . [http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf]